AD-A231 378

NAS (HOST/ARTS IIIA) to VME Modem Interface ATC Interface Hardware Manual

DTIC
ELECTE
FEB 26 1991
S B D

Leo J. Wapelhorst

October 1990

DOT/FAA/CT-TN90/46

This document is available to the U.S. public through the National Technical Information Service, Springfield, Virginia 22161.

US Department of Transportation
Federal Aviation Administration

Technical Center
Atlantic City International Airport, N J 08405

91 2 20 001

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/CT-TN90/46		

4. Title and Subtitle	5. Report Date
NAS (HOST/ARTS) IIIA TO VME MODEM INTERFACE ATC INTERFACE HARDWARE MANUAL	October 1990
	6. Performing Organization Code ACD-320
7. Author(s) Leo J. Wapelhorst	8. Performing Organization Report No. DOT/FAA/CT-TN90/46
9. Performing Organization Name and Address Department of Transportation FAA Technical Center Atlantic City International Airport Atlantic City, New Jersey 08405	10. Work Unit No. (TRAIS)
	11. Contract or Grant No. T2001F
12. Sponsoring Agency Name and Address Department of Transportation FAA Technical Center Atlantic City International Airport ATlantic City, New Jersey 08405	13. Type of Report and Period Covered Technical Note
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

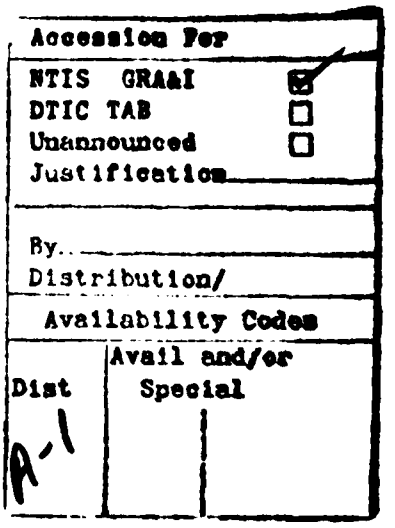
This document is reference material for personnel using the National Airspace System (NAS) (HOST or ARTS IIIA) Air Traffic Control (ATC) Interface Subsystem. It was originally developed to be part of the Data Link Test and Analysis System (DATAS) in order to provide an interface between the NAS and the Ground Data Link Processor (GDLP).

17. Key Words	18. Distribution Statement
ARTS III DATAS HOST NAS	This document is available to the U.S. public through the National Technical Information Service, Springfield, VA. 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	120	

TABLE OF CONTENTS

	Page
EXECUTIVE SUMMARY	v
GENERAL DESCRIPTION	1
RELATED DOCUMENTS	3
HARDWARE CONTROL	3
Hardware Diagnostic Menu (L)	3
Card Configuration Menu (J Menu)	8
Memory Diagnostics	9
MEMORY MAP OF ATC2 INTERFACE	9
Address Decoding	10
Interrupt Processing	11
Input Clock Control	14
TRANSMIT DATA PROCESSING	15
RECEIVE DATA PROCESSING	17
Receive Data Sync Detection	17
Receive Status Logic	19
Receive Parity Logic	24
EOM Detection	24
Receive Memory Address Control	24
APPENDICES	
A - Transmit Data Sequences	
B - Receive Data Sequences	
C - PAL Source Code	



LIST OF ILLUSTRATIONS

Figure		Page
1	ATC Interface Block Diagram	2
2	ATC Interface Card (2 Sheets)	4
3	ATC Interface Schematic with Board Layout (2 Sheets)	6
4	ATC Interrupt Logic	12
5	ATC Interrupt State Diagram	13
6	ATC Transmit Control State Diagram	16
7	ATC Sync Detector State Diagram	18
8	ATC Bit Counter State Diagram	20
9	ATC Receive Status State Diagram	22
10	ATC Reply Memory Address Control Block Diagram	25

EXECUTIVE SUMMARY

This document provides reference material for personnel using the air traffic control (ATC) interface between the National Airspace System (NAS) and the Data Link Test Bed. This interface card can be used on any Versa Module Extended (VME) based computer system to provide an interface between that system and the NAS. It was originally designed to be part of the Data Link Test Analysis System (DATAS) in its role to provide interfaces to any part of the NAS components used for Data Link.

This document contains a detailed description of the hardware of the ATC Interface card. It includes schematics, Pal Source code, diagnostic descriptions, and should be sufficient for hardware personnel to become familiar with the operation of the card.

GENERAL DESCRIPTION

The Air Traffic Control (ATC) Interface card is a Versa Module Extended (VME) bus compatible module which provides an interface between the NAS (HOST or ARTS) system and the VAX-750. This card communicates with those systems via the synchronous modems of the LSSS. Intermediate processing of the data is provided by the 68020 processor of the VME system. This card can be used in the Data Link Test Analysis System (DATAS) or any other VME based 68020 system.

The ATC interface card has extensive "built in" diagnostic capability which can be controlled by the 68020. It has the ability to use either external or internally generated modem clocks (between 300 and 19,200 baud) for data processing. Configuration of the entire card is controlled via the 68020. Signals can be "looped back" for monitoring; sync detection logic, transmit parity generation logic and interrupts can be disabled. It also contains a mechanism for storing all incoming data (including the parity bits) if desired. The use of this capability makes it possible for two cards to reside in one VME chassis where one unit can be configured to monitor the performance of the other if desired.

The logic of the ATC Interface card is almost entirely in programmable logic and can be easily changed if desired.

Figure 1 is a block diagram of the ATC Interface card. Communication with the VME system is via A32/D08 standard protocol. The interface card is assigned a particular block of addresses for this purpose. It monitors the address lines as well as the address modifier lines and responds when one of the assigned addresses is accessed by the processor. Its transmit data is received from the processor and stored in "dual port" random access memory (RAM) so that both the card and processor can access the same locations. Control logic for configuration of the card is received in the same manner. When the processor desires to transmit a message to an external system, it loads the Transmit memory with the message and the control latches with the length of the message and issues a "start" command. When the card has completed the assigned transmission, it sends an interrupt request to the processor to indicate completion of the task.

The ATC Interface card receives its data from the external device (or internally if in the loop back mode), stores it in the Receive Memory and notifies the processor via an interrupt whenever a valid End of Message (EOM) is received or a parity error occurs. The card can also be configured to store all incoming data after a specified number of consecutive zeros occurs (controlled via 68020). If the receive sync detection logic is enabled, the data will be "overwritten" with the input data if a valid sync message (either 17 or 18 consecutive zeros followed by a 1) is detected. If no valid sync occurs (or sync detection is disabled) an

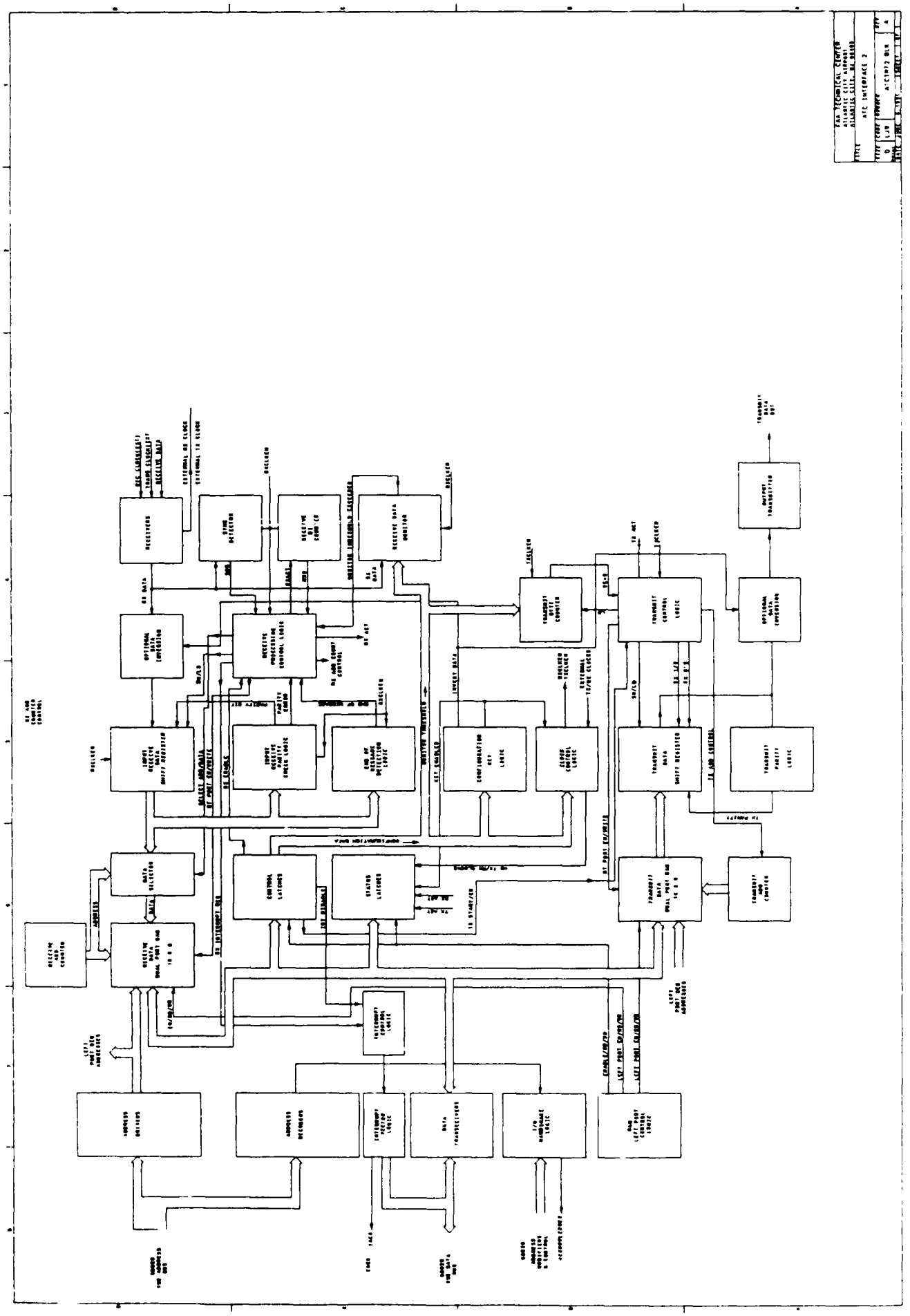


FIGURE 1. ATC INTERFACE BLOCK DIAGRAM

interrupt is issued to the processor when 128 bytes have been stored. The receive data are stored in memory blocks of 128 bytes. Each block is assigned a separate interrupt vector, so that the processor is apprised of the block which contains the end of the message when it receives the interrupt. These blocks are used in sequence so that the processor need not keep up in real time with the reading of the data.

RELATED DOCUMENTS

NAS-MD-601:	This is the interface control document for NAS EN ROUTE - ARTS IIIA.
atcint2.icd:	This document defines the addresses and all control bits.
atc2.s:	This document is the source code for the PALs of the board and is included as appendix C.
atci2_1.dwg:	This is sheet 1 of the schematic and is shown in this document as figure 2, sheet 1.
atci2_2.dwg:	This is sheet 2 of the schematic and is shown in this document as figure 2, sheet 2.
atc2dr.dwg:	This is sheet 1 of the schematic but also includes the board layout and is shown in this document as figure 3.
atc2.pal:	This file shows the file information for all the PALs (U#, board location, PAL type, source file name, jedec file name and date, source file location, and various comments about the function. A copy of this file is included as figure 3, sheet 2.

HARDWARE CONTROL

All of the hardware controls are exercised via menu driven selections from the system console. When the system is initialized (the program is called **nas** and is in user directory **100.nas**), it comes up with a top level menu. This is the menu which is used by the system personnel when actually working with the NAS. There are two submenus for diagnostic purposes: (1) the Hardware Diagnostic menu which is accessed by entering L from the Top Level and (2) the Software Diagnostic menu which is accessed by entering G from the top level. Only the Hardware menu will be discussed here.

HARDWARE DIAGNOSTIC MENU (L).

The Hardware Diagnostic menu allows the following selections:

1 - **Display RX buffer** (a submenu allows selection of choice)
2 - **Display TX buffer**
3 - Not used
4 - Transmit 8 **byte test message** (ends with parity error)
5 - 1/**sec XMT test** (1 to 256 bytes)
6 - 1/**sec L/B test**

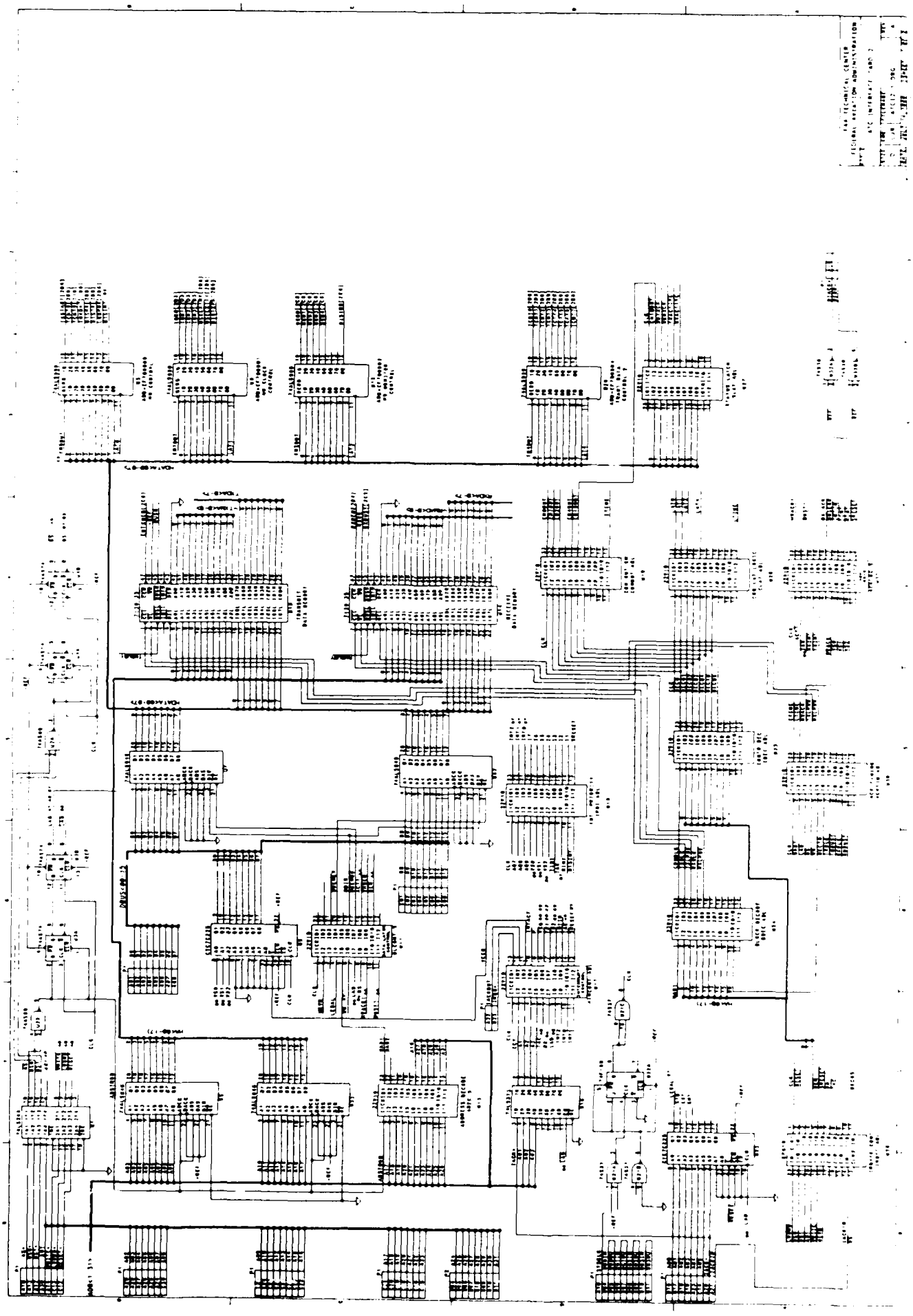


FIGURE 2. AIC INTERFACE CARD (SHEET 1 OF 2)

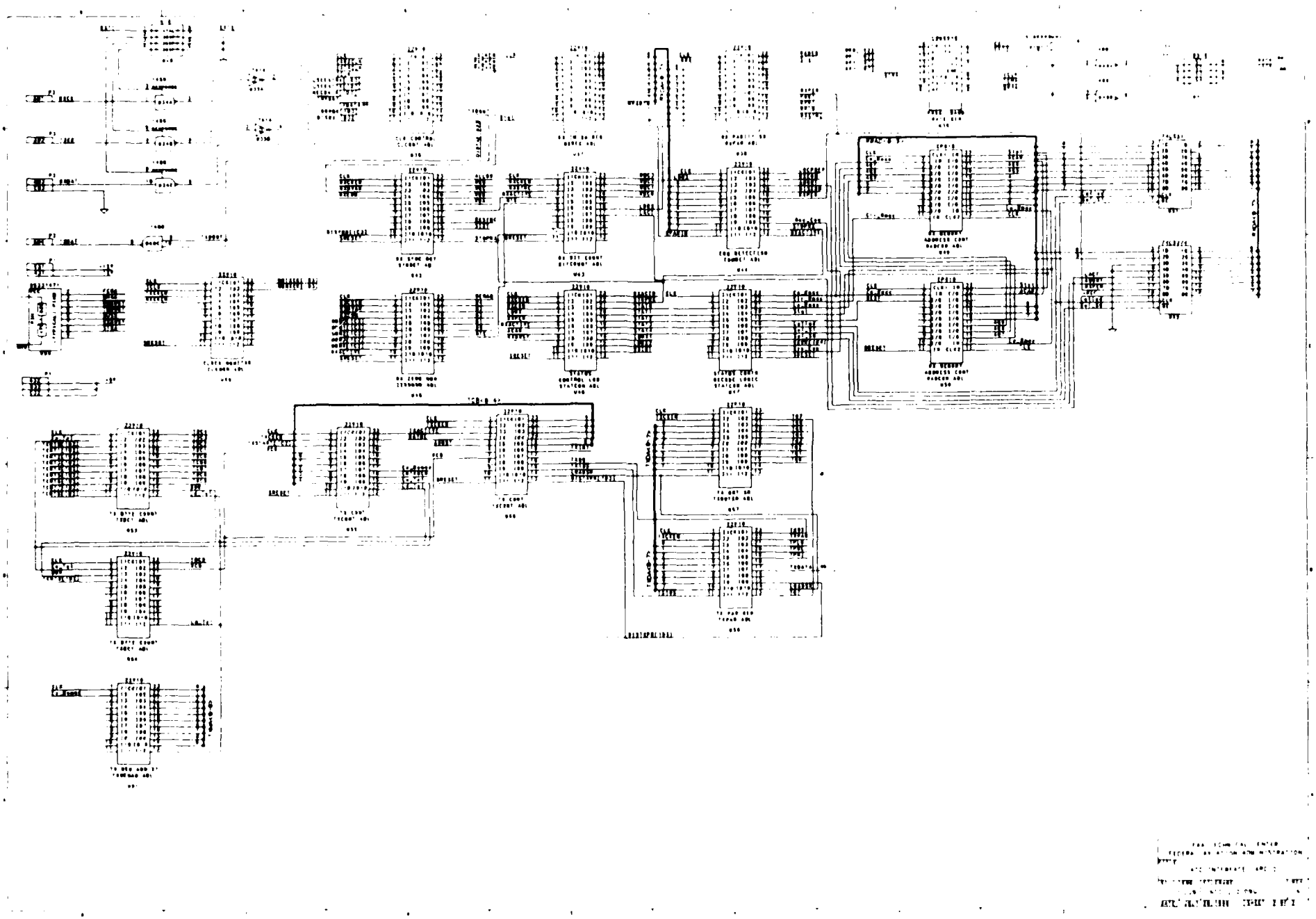


FIGURE 2. ATC INTERFACE CARD (SHEET 2 OF 2)

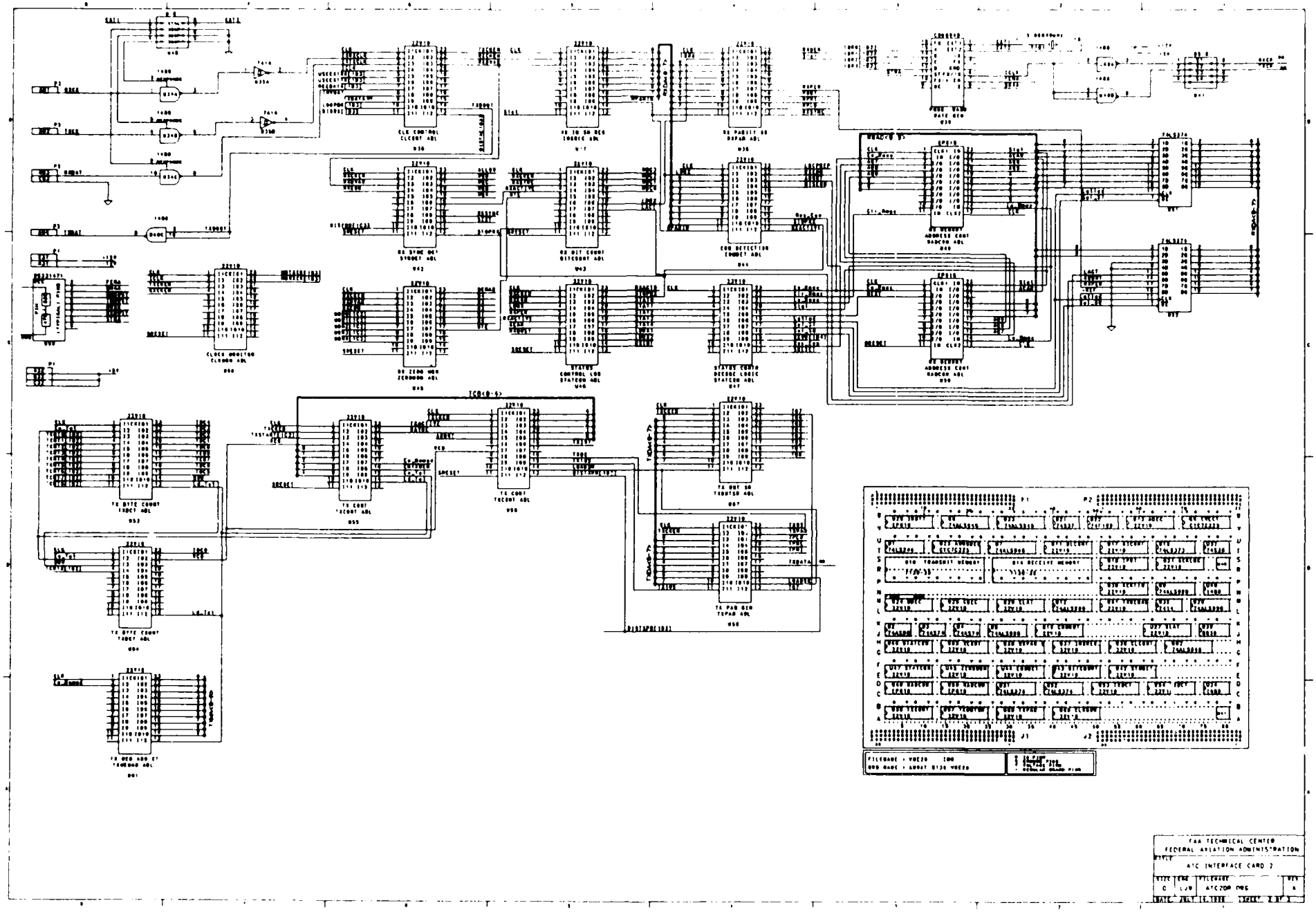


FIGURE 3. ATC INTERFACE SCHEMATIC WITH BOARD LAYOUT (SHEET 1 OF 2)

PARTS DATABASE REPORT DATE: 10-05-1990 PAGE: 1

ATC INTERFACE PAL CATALOG

Part Ref	LOC	PALTYPE	SOURCE FILE	JED	DATE	DIRECTORY	COMMENTS	SPARE
U11	T38	22V10	DLCONT.ABL	U11	06/29/0	LAB E:\DAI\ATC	DATA LATCH CONTROL LOGIC	
U13	V57	22V10	ADEC.ABL	U13	06/27/0	LAB E:\DAI\ATC	VME ADDRESS DECODE LOGIC	
U17	T51	22V10	AINCONT.ABL	U17	06/27/0	LAB E:\DAI\ATC	INTERRUPT CONTROL LOGIC	
U18	R51	22V10	IPRI.ABL	U18B	07/07/0	LAB E:\DAI\ATC	INTERRUPT PRIORITY LOGIC	
U19	J36	22V10	CONOUT.ABL	U19	06/19/0	LAB E:\DAI\ATC	CONTROL LATCH OUTPUT ENABLE LOGIC	
U23	T19	CYC7C225	AMODEC.ATC	NONE	06/28/0	LAB E:\DAI\ATC	VME BUS ADDRESS MODIFIER DECODER	
U24	L1	22V10	BDEC.ABL	U24	06/19/0	LAB E:\DAI\ATC	ADDRESS BLOCK DECODE LOGIC	
U25	L14	22V10	CDEC.ABL	U25	06/19/0	LAB E:\DAI\ATC	CONTROL ADDRESS DECODE LOGIC	
U26	L27	22V10	CLAT.ABL	U26	06/19/0	LAB E:\DAI\ATC	CONTROL LATCH LOAD LOGIC	
U27	J61	22V10	SLAT.ABL	U27	06/19/0	LAB E:\DAI\ATC	HARDWARE STATUS LATCH	
U28	V1	EP610	INBYT.ABL	U28	06/28/0	LAB E:\DAI\ATC	INPUT BYTE CONTROL LOGIC	
U30	N51	22V10	ACKTIM.ABL	U30	06/18/0	LAB E:\DAI\ATC	VME ACKNOWLEDGE TIMING LOGIC	
U31	R64	22V10	ACKLOG.ABL	U31B	07/07/0	LAB E:\DAI\ATC	VME ACKNOWLEDGE LOGIC	
U36	G53	22V10	CLCONT.ABL	U36B	10/04/0	LAB E:\DAI\ATC	SYSTEM CLOCK CONTROL LOGIC	
U37	G40	22V10	INSREG.ABL	U37	06/20/0	LAB E:\DAI\ATC	RECEIVE DATA INPUT SHIFT REGISTER	
U38	G27	22V10	RXPAR.ABL	U38	07/06/0	LAB E:\DAI\ATC	RECEIVE DATA PARITY LOGIC	
U42	E53	22V10	SYNDET.ABL	U42	07/04/0	LAB E:\DAI\ATC	RECEIVE DATA SYNC DETECTOR	
U43	E40	22V10	BITCOUNT.AB	U43	07/09/0	LAB E:\DAI\ATC	RECEIVE DATA BIT COUNTER	
U44	E27	22V10	EOMDET.ABL	U44	07/13/0	LAB E:\DAI\ATC	RECEIVE DATA END OF MESSAGE DET	
U45	E14	22V10	ZEROMON.ABL	U45	07/19/0	LAB E:\DAI\ATC	RECEIVE DATA CONSECUTIVE ZERO MON	
U46	G1	22V10	STATCON.ABL	U46A	07/18/0	LAB E:\DAI\A	RECEIVE DATA STATUS CONTROL	
U47	E1	22V10	STATCON.ABL	U47	07/18/0	LAB E:\DAI\A	RECEIVE DATA STATUS CONTROL	
U49	C1	EP610	RADCON.ABL	U49	07/06/0	LAB E:\DAI\ATC	RECEIVE DATA MEMORY ADDRESS CONTROL	
U50	C14	EP610	RADCON.ABL	U50	07/06/0	LAB E:\DAI\ATC	RECEIVE DATA MEMORY ADDRESS CONTROL	
U53	C49	22V10	TXBCT.ABL	U53	06/22/0	LAB E:\DAI\ATC	TRANSMIT DATA BYTE COUNTER	
U54	C62	22V10	TXBCT.ABL	U54	06/22/0	LAB E:\DAI\ATC	TRANSMIT DATA BYTE COUNTER	
U55	G14	22V10	TXCONT.ABL	U55	07/17/0	LAB E:\DAI\ATC	TRANSMIT DATA CONTROL LOGIC	
U56	A1	22V10	TXCONT.ABL	U56	07/17/0	LAB E:\DAI\ATC	TRANSMIT DATA CONTROL LOGIC	
U57	A14	22V10	TXOUTSR.ABL	U57	07/18/0	LAB E:\DAI\ATC	TRANSMIT DATA OUTPUT SHIFT REGISTER	
U58	A27	22V10	TXPAR.ABL	U58	07/19/0	LAB E:\DAI\ATC	TRANSMIT DATA PARITY GENERATOR	
U60	A40	22V10	CLKMON.AB	U60	06/22/0	LAB E:\DAI\ATC	EXTERNAL CLOCK MONITOR	
U61	L51	22V10	TXMEMAD.ABL	U61	07/03/0	LAB E:\DAI\ATC	TRANSMIT MEMORY ADDRESS COUNTER	
U9	V77	CYC7C225	IVEC.ATC	NONE	06/28/0	LAB E:\DAI\ATC	INTERRUPT VECTOR SELECTION PROM	

FIGURE 3. ATC INTERFACE SCHEMATIC WITH BOARD LAYOUT (SHEET 2 OF 2)

7 - Not used
8 - **8 Byte EOM test message** (transmitted once)
9 - **Compose Test message** (enter your own message)
A - **View composed test message**
B - **XMT composed test message**
C - **XMT composed test message 1/sec**
J - **Card Configuration menu**
K - **Rewrite Screen**
L - **Go to top level menu**

CARD CONFIGURATION MENU (J MENU).

This menu reads the **Hardware Status Control register** and interprets the contents so that the configuration of the hardware is easily seen. The status of the following parameters is given on the menu:

Interrupts	Enable/Disable
Loop Back	Enable/Disable
XMT data	Enable/Disable
RCV data	Enable/Disable
Invert data	Enable/Disable
RX sync	Enable/Disable
XMT Logic	Enable/Disable
# of Consecutive 0's	Value

It also reads the **Hardware Status Register** and reports the status of the following hardware parameters:

Status Register	Contents
TX Clock	Present/No TX clock
RX Clock	Present/No RX clock
Xmt Logic	Active/Inactive
RCV Logic	Active/Inactive

The J Menu also allows the selection of the configuration parameters from the following menu:

1 - Reset Card
2 - Enable Operation
3 - Disable Operation
4 - Set Baud Rate (300 to 19200 for internal clock)
5 - Enable Interrupts
6 - Disable Interrupts
7 - Enable Loop back
8 - Disable Loop Back
9 - Enable XMT data
A - Disable XMT data
B - Enable RCV data
C - Disable RCV data
D - Enable Invert data
E - Disable Invert Data

F - USE EXT XMT clock
G - UNUSE EXT XMT clock
H - USE EXT RCV clock
I - UNUSE EXT RCV clock
J - Enable RCV Sync Detection
K - Disable RCV Sync Detection
L - Set # consecutive sync 0's
M - Enable XMT Logic
N - Disable XMT Logic
R - Set XMT Test Conditions
S - Leo Test Mode Menu (hardware diagnostics)
T - George Test Mode Menu (software diagnostics)
U - Top Level Menu

MEMORY DIAGNOSTICS.

The memory diagnostics are run via a program called **nasmtest**. This program is menu driven and provides the following capabilities:

1. The ability to address any address within the system (not limited to those assigned to the ATC interface.

2. The ability to perform a "write - read - read" cycle on any address. Data are variable.

3. The ability to perform a constant "read" of any address. This is mainly for troubleshooting purposes.

4. The ability to perform a constant "write" of any address. This is also for troubleshooting purposes.

5. A function which verifies "write-read" data on one address. Data is variable but not selectable.

6. A function which verifies "write-read" data of 0 to 255 on a specified number of addresses.

7. A function which shifts a single 1 pattern to the left and verifies proper performance.

8. A function which shifts a single 0 pattern to the left and verifies proper performance.

9. The ability to "mask out" certain bits so that the processor does not check them. This function is used mainly to eliminate going to the screen with the errors as it slows down the prf of the functions considerably.

MEMORY MAP OF ATC2 INTERFACE

The memory of the ATC2 interface hardware will by organized as follows:

The memory of the ATC interface will be assigned to the addresses in the EF700000 block of system memory. It will be organized so that several interface cards can reside in the same unit, with the only difference being the addresses of each unit (one Pal difference). Assignments will be as follows:

```
EF700000-EF70FFFF   INTERFACE UNIT #1
EF710000-EF71FFFF   INTERFACE UNIT #2
EF720000-EF72FFFF   INTERFACE UNIT #3
EF730000-EF73FFFF   INTERFACE UNIT #4
```

The breakdown of memory allocation within each block is as follows:

```
EF700000  Hardware control Register
EF700001  Clock Contol Register
EF700002  Consecutive Zero Monitor Control
EF700003  No Longer Used
EF700004  Transmit Data Byte Count (8 lsbs)
EF700005  Software System Enable (1=enable, 0=disable)
EF700010  Hardware Status Word (Read Only)
EF700011
   TO
EF700FFF  NOT USED
EF701000
   TO
EF7013FF  Receive Data Memory
EF701400
   TO
EF701FFF  NOT USED
EF702000
   TO
EF7023FF  Transmit Data Memory
EF702400
   TO
EF70FFFF  NOT USED
```

All memory except for the Hardware Status word (EF700010) is **dual port** and can be written or read by the computer interface.

ADDRESS DECODING.

The ATC Interface card is set up as an A32 (32 address bits)/D08 (8 data bits) interface. It monitors the address modifier lines on the VME bus and responds to only address modifiers of 09, 08, 0D, and 0E which are valid only for A32/D08 systems. The two least significant bits of address are defined by the following truth table.

DS1	DS0	A01	LWORDLSB ADD
0	X	0	X0
0	X	1	X2
X	0	0	X1
X	0	1	X3

The address decode function is divided between several PALs on the interface card. ADEC.ABL (U13) decodes address bits 16 to 31 to break the decodes into 64k blocks. When the appropriate 64k block is accessed which matches that assigned to the particular interface card, it produces MAD1 to signify that fact. BDEC.ABL (U24) further breaks down the decode to either CONTROL, TRANSMIT MEMORY, or RECEIVE MEMORY. Its outputs are passed on to the final decode pal CDEC.ABL (U25) which produces the appropriate enables for the logic of the particular areas of the interface logic by using these inputs and the 8 Lsb's of the VME address. (The two Lsb's are produced by INBYT.ABL (U28) to decode the four signals which are used by the VME to address the two lowest address bytes (LWORD, DS0, DS1, and A01). In order to prevent right and left port simultaneous accesses to the same memory address, the right port (hardware side) is only enabled when an active transmit or receive cycle is in progress.

The acknowledge handshake logic for the VME bus is handled by PALs U30 (ACKTIM.ABL) and U31 (ACKLOG.ABL). These PALs use the data and address strobes from the VME bus along with the "write" control signals to produce the acknowledges required to satisfy the handshake protocol of the VME.

INTERRUPT PROCESSING.

All of the hardware interrupts to the 68020 are at system interrupt level 6. Figure 4 is a block diagram of that interrupt logic. There are eight different interrupt vectors. One is assigned to the transmit interrupt and the other seven are assigned to the receive data buffers. The buffer which contains the last valid receive data is used to generate the proper vector so that the processor can properly locate the latest data.

AINCONT.ABL (U17) controls the interrupt handshaking with the 68020. It receives the requests via INT0-2 from the priority PAL (IPRI.ABL). It accepts higher level interrupts until the end of state 1. It then freezes at that vector address until state 7 when it is reset. It furnishes the priority PAL with IAK which resets the interrupt request. The drawing of the state diagram is shown in figure 5.

IPRI.ABL (U18) stores interrupt requests from the system and passes them along to U17 according to priority (IRQ7 is the highest priority). Transmit interrupts (TINT) have priority over receive (RINT). The interrupt requests are active high and remain set

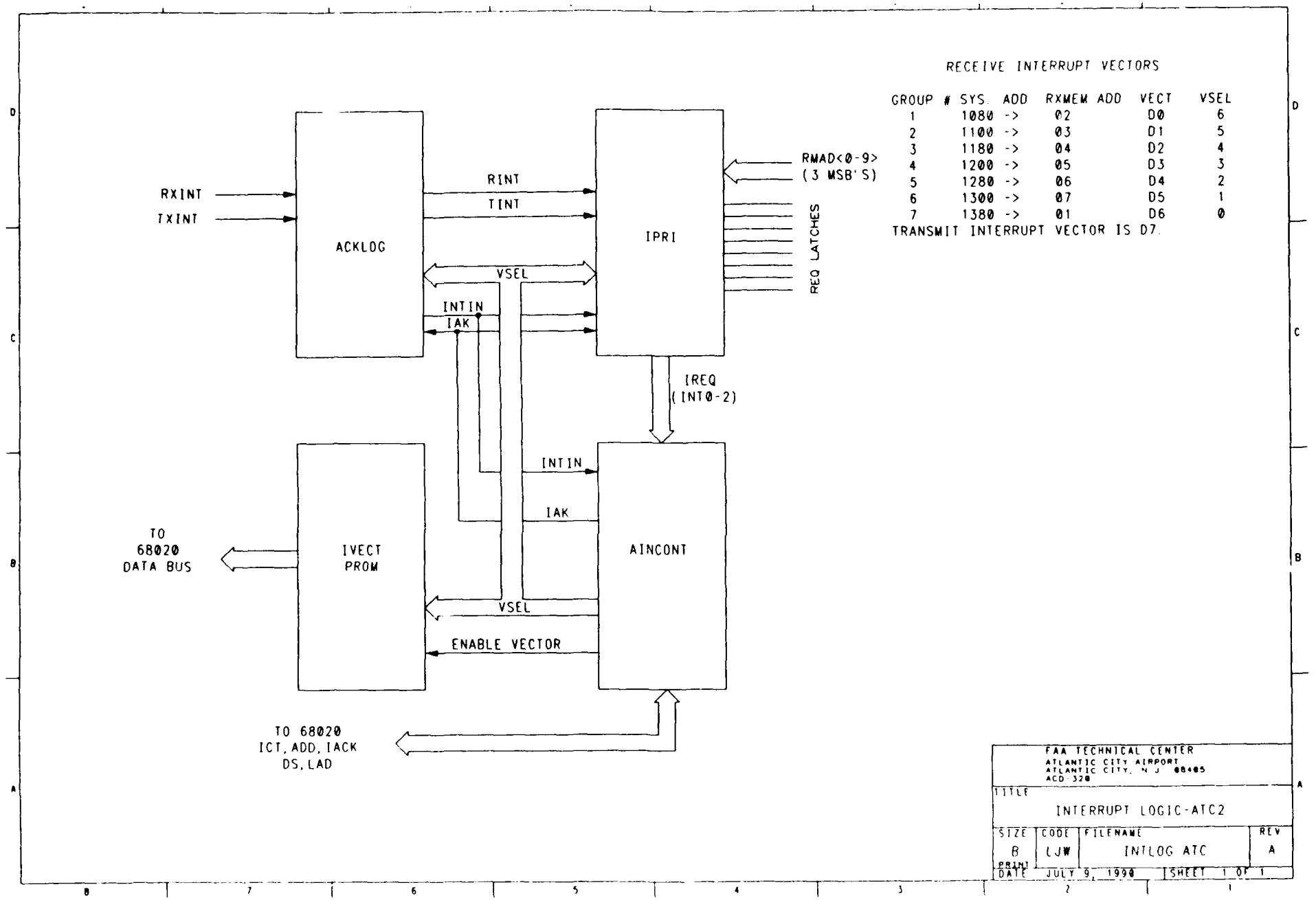


FIGURE 4. ATC INTERRUPT LOGIC

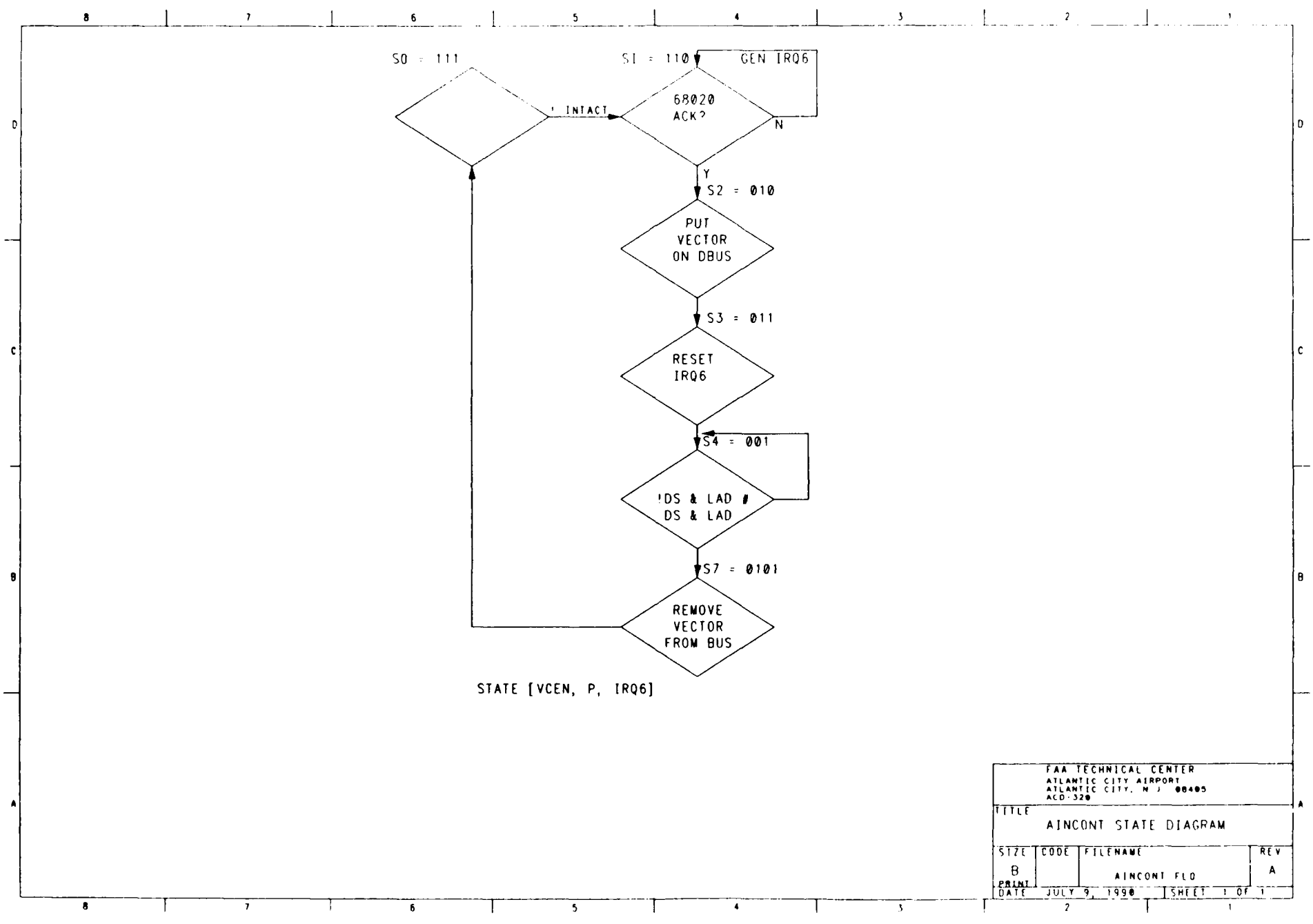


FIGURE 5. ATC INTERRUPT STATE DIAGRAM

until the interrupt is serviced. The vector request is an ACTIVE LOW three-bit field (IRQA,IRQB,IRQC). The outputs are also active low (INT2,INT1,INT0). All interrupts are cleared when a RESET is received. A separate vector is generated for each block of addresses of receive data.

The interrupt vectors for the various levels of interrupts are defined as follows:

Rec. Data Group 1	Vec.Prom Add.	Vector
1	006	D0
2	005	D1
3	004	D2
4	003	D3
5	002	D4
6	001	D5
7	000	D6

The vector for the transmit data interrupt is D7 and it is at address 007.

The requests for interrupts come from the transmit and receive data logic in the form of RXINT and TXINT. These signals are processed by the ACKLOG.ABL and passed on to the interrupt processing logic. TXINT is produced when the designated number of bytes have been transmitted by the hardware. In this manner, the computer is notified of the completion of the assigned task. The receive interrupt request (RXINT) comes to ACKLOG.ABL from the receive status logic (STATCON.ABL). See the description of the status logic for details.

INPUT CLOCK CONTROL.

The serial input clock control is handled by PAL U36B. This PAL is controlled by the following signals:

USEXTRX - Selects either the internally generated or external clock for use of receive logic.
USEXTTX - Selects either the internally generated or external clock for use of transmit logic.
LOOPBK - Uses either incoming RECDATA or output transmit data for input receive data.
INVDAT - Provides the option of inverting both transmit and receive data.

This module contains logic which effectively filters glitches on both transmit and receive clocks. The rise times of these signals can be several microseconds and the hysteresis provided by the line receivers is sometimes not sufficient to eliminate small glitches. Whenever the input clocks are high, a counter is enabled to count up to seven (where it remains until the input goes back low). The counter is reset every time the input clock goes low. Thus, if a

glitch occurs at the count of four or five, it merely restarts the counter. The external transmit logic is enabled by TXCKEN which occurs at the positive going edge of the input clock. The external receive logic is enabled by RXCKEN which occurs at the negative edge of the input clock. Both occur at a count of six of their respective counters. Either the external clocks or the internally generated clock (300 - 19200) can be selected by the input control signals.

TRANSMIT DATA PROCESSING

The transmit data processing logic is controlled via the Transmit Control logic of U55 and U56. This module is actually a state controller whose state diagram is shown in figure 6. If the processor desires to transmit a message, it first stores the transmit data in the Transmit Memory, U10. It stores the number of bytes to transfer in latches U5 and U20. U5 also contains the "transmit start" signal (TXSTART) which occurs on the transition of bit 0 from a 0 to a 1. This event sets TXACTIVE which enables the Transmit Control State controller to run. It moves one state at the Lead Edge of each Transmit Clock. State S1 starts the generation of the sync pattern. S2 enables the load of the byte count into the byte counter U53, U54. The transmit output continues to be all zeros until state S17 is reached, which generates a 1 for the sync message. S18 enables the loading of the shift register with the first byte of output data. S19 checks for a "word count=0" and since this is not the case, the controller goes to state S20 which enables the memory address counter to increment to the next byte. The next state, S21 enables the byte counter to decrement. The state controller then goes through states S22, S23, S24, and S25 while shifting out the data from the transmit shift register U57 via the transmit parity generator U58. At state S26 a check is made to see if the transmit parity generation logic is enabled (this requires a nine state loop - 8 data bits plus parity). If the logic is enabled, it goes to state S27 which loads the next byte into the shift register and transmits the parity bit out the serial transmit port. If the logic is disabled, it skips state S27 and goes to S19 to check for a word count of zero. The loop continues until a word count of zero condition is reached. At this time TXACTIVE is reset, a transmit interrupt request is issued, and the state controller goes to the IDLE state S0.

Appendix A is a file collected via a logic analyzer which shows two transmit sequences. The first is with the transmit logic enabled (pages A-1 to A-3) and the second is with the transmit logic disabled (pages A-4 to A-6). Both sets of data contain comments which flag significant events on any particular state.

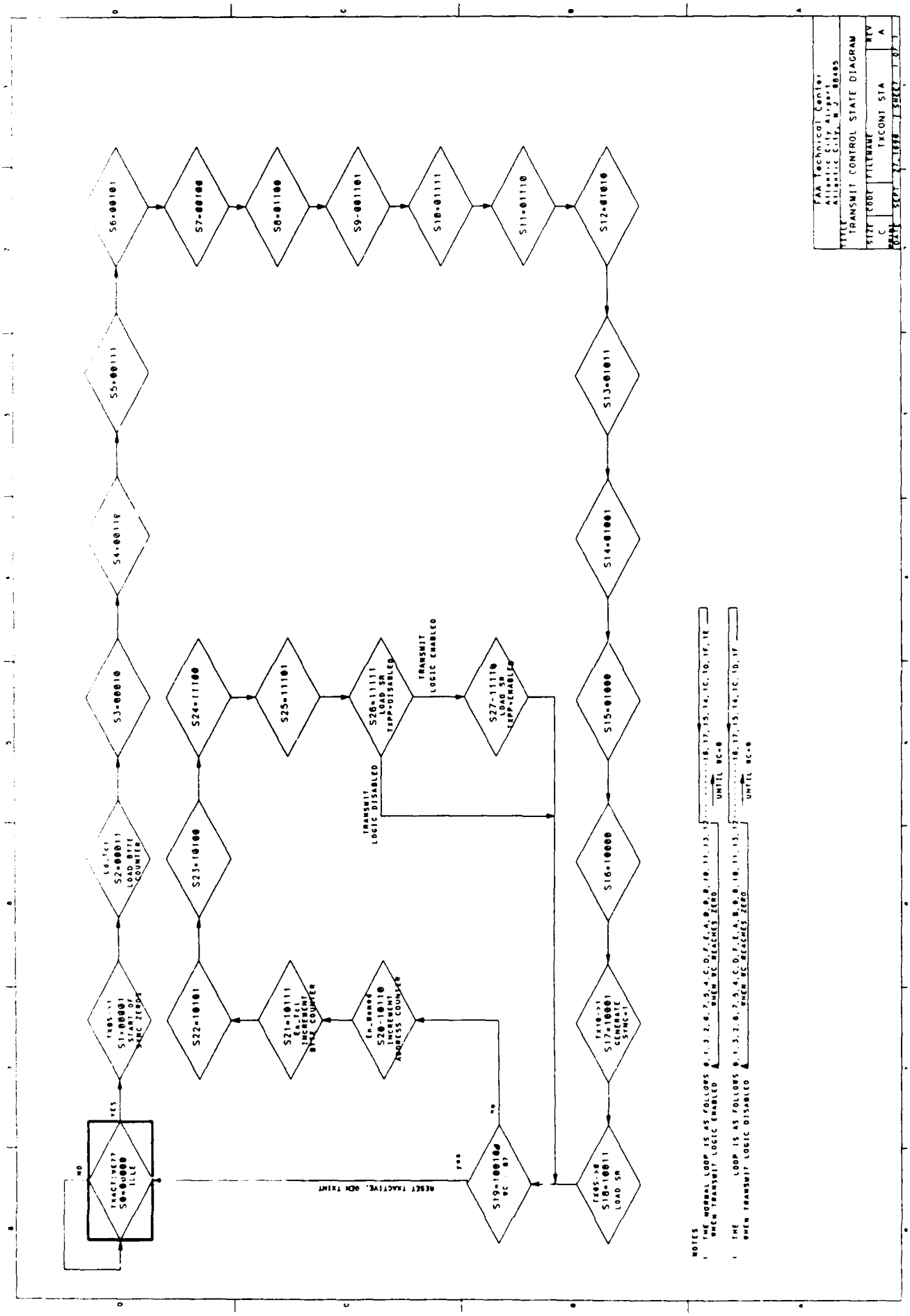


FIGURE 6. ATC TRANSMIT CONTROL STATE DIAGRAM

RECEIVE DATA PROCESSING

Many features are included for analysis of system operation when peculiarities occur. The normal sync pattern for the incoming data is either 17 or 18 zeros (depending on the level of the idle character when the sync pattern starts) followed by a 1. A common problem is the missing of a sync character because a 1 was picked up by the receive logic when a 0 was actually transmitted on the other end. The ATC interface contains logic to monitor the number of consecutive zeros prior to storage of data. When the threshold (controlled via the processor and address EF700002) is exceeded, all incoming data (including the parity bit) is stored in RAM so that it can be read by the processor. If an actual sync pattern is later detected prior to the storage of 128 bytes, the stored data is **over written** with the real message data. In this mode the parity bits are stripped off by the receive processing logic and only the message is stored.

Three sets of data are included as appendix B. They show receive logic state sequences for three different events.

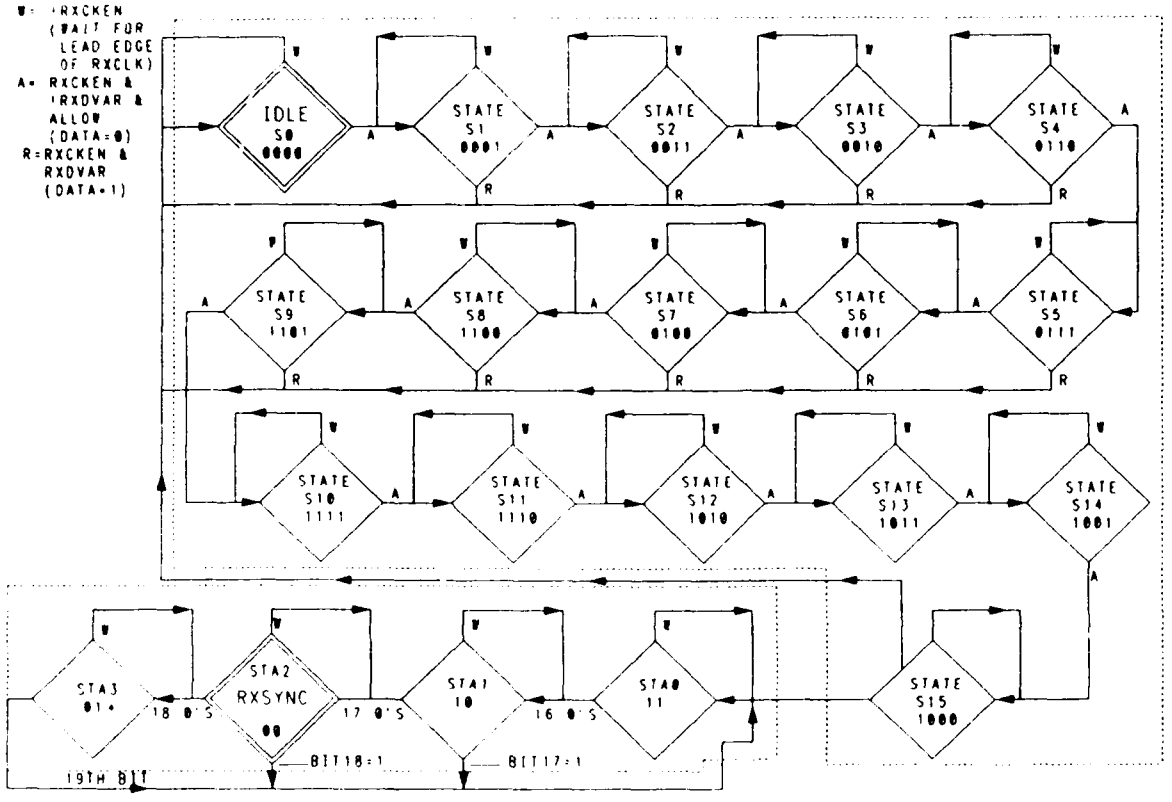
1. Pages B-1 to B-4 show the processing when the Receive Sync detection logic was **DISABLED** and a standard 8 byte message was received. Significant events are commented as the sequence is traced from sync occurrence and the exceeding of the MTE threshold until the end of the message occurs. This sequence illustrates the storage of all incoming data, including parity. The use of this data along with the state diagrams will make it easy to understand the receive logic.

2. Pages B-5 to B-7 show the same sequence as described above, but with Receive Sync Detection logic **ENABLED**. This shows the storage of only the real input data, with the parity bits stripped off by the hardware.

3. Pages B-8 to B-10 show the actual storage of data cycles. These sequences are actually performed by the Status Logic, which controls the storage of all Receive data. These sequences are in single 100 ns clock increments, unlike 1 and 2 above which were one clock sequence per Receive Clock (RXCK). These three sequences show a normal Memory storage cycle, a Status storage cycle, and a Reset cycle which occurs when sync is detected.

RECEIVE DATA SYNC DETECTION.

Sync detection is controlled by U42 (SYNDET.ABL). This logic searches for the sync pattern of either 17 or 18 zeros followed by a 1 to identify the start of a message. U42 is comprised of two state controllers which use RXCKEN and RXDVAR as the control inputs for the state controllers. Equations are given in SYNDET.ABL and the state diagram is shown on figure 7.



* STATE CONTROLLER
STA [RXSYNC,SIXT]

NOTES 1 SYNDET ALSO PRODUCES 'ALLOW' AND 'RXACTIVE'
2 ALLOW IS SET WHENEVER THE DATA BIT = 1
3 RXACT.VE IS SET ON THE '1' FOLLOWING RXSYNC IF IT IS THE 18TH OR 19TH BIT AND REMAINS SET UNTIL RXEOM IS RECEIVED

FAA TECHNICAL CENTER ATLANTIC CITY AIRPORT ATLANTIC CITY, N.J. 08405 ACD-320			
TITLE: SYNC DETECTOR STATE DIAGRAM			
SIZE: B	CODE: I.D.W.	FILENAME: SYNDET STA	REV: A
PRINT DATE: JUNE 26, 1990		SHEET 1 OF 1	

FIGURE 7. ATC SYNC DETECTOR STATE DIAGRAM

The enable signal for the state controllers is ALLOW, which is activated any time a 1 is detected in the input data stream. It remains active until a string of 19 consecutive zeros disables the sync detection logic until another 1 is detected.

The first of the two state controllers in U42 (outputs QA,QB,QC and QD)detects 16 consecutive zeros. It is reset to state S0 when any input data bit is a 1. On the 16th zero, output SIXT goes low. On the 17th consecutive zero, RXSYNC is generated. If the 18th data bit is a 1, RXSYNC is terminated and RXACTIVE is generated. If the 18th data bit is a 0, the controller waits for the 19th bit. If the 19th bit is a 1, RXSYNC is terminated and RXACTIVE is generated (this timing is also shown on figure 1). If the data bit is another zero, ALLOW is disabled and the process must begin anew after another 1 has been received. All of the above cases are covered in the "test_vectors" portion of the PAL source code file SYNDET.ABL.

Receive data storage control is under the control of U43, BITCOUNT.ABL - This PAL contains the logic to control storage of receive data in memory. When the "consecutive zero threshold" is exceeded (MTE), incoming data are stored in RAM as it comes in (every eight bits). If sync is detected, the address counter is reset and data is stored every nine bits as parity is not stored in memory for real data.

This is accomplished in the following manner: A state controller which has 10 states (see figure 8) is in the IDLE state until MTE occurs. When this happens, the state controller goes around a loop as follows: S0(IDLE) jumps to S2-S3-S4-S5-S6-S7-S8-S9-S2-S3, etc. Data are written into RAM on S9. If sync occurs, the state controller goes to state S1 and also loops back to S1 instead of S2 making the loop equal to nine data bits. Data are still stored at state S9 by this PAL generating LDRX. The actual storage process is controlled by another PAL. This PAL also produces LACT which is a signal which lasts from the time MTE occurs until a real sync is detected (RXSYNC). If RXSYNC is detected before 128 bytes are stored, LACT is reset, the loop changes to nine bits, and the data previously written in RAM are written over with the actual message data. If no sync is detected, the storage of data ends after 128 bits and the status logic strobes the value of LACT to indicate to the computer that no valid start of message was detected, but the threshold was exceeded.

RECEIVE STATUS LOGIC.

The receive data are stored in RAM in blocks of 128 bytes. Each group has a 4-byte status word associated with it to identify some of its characteristics for the processor. The address assignments of these status blocks is shown below:

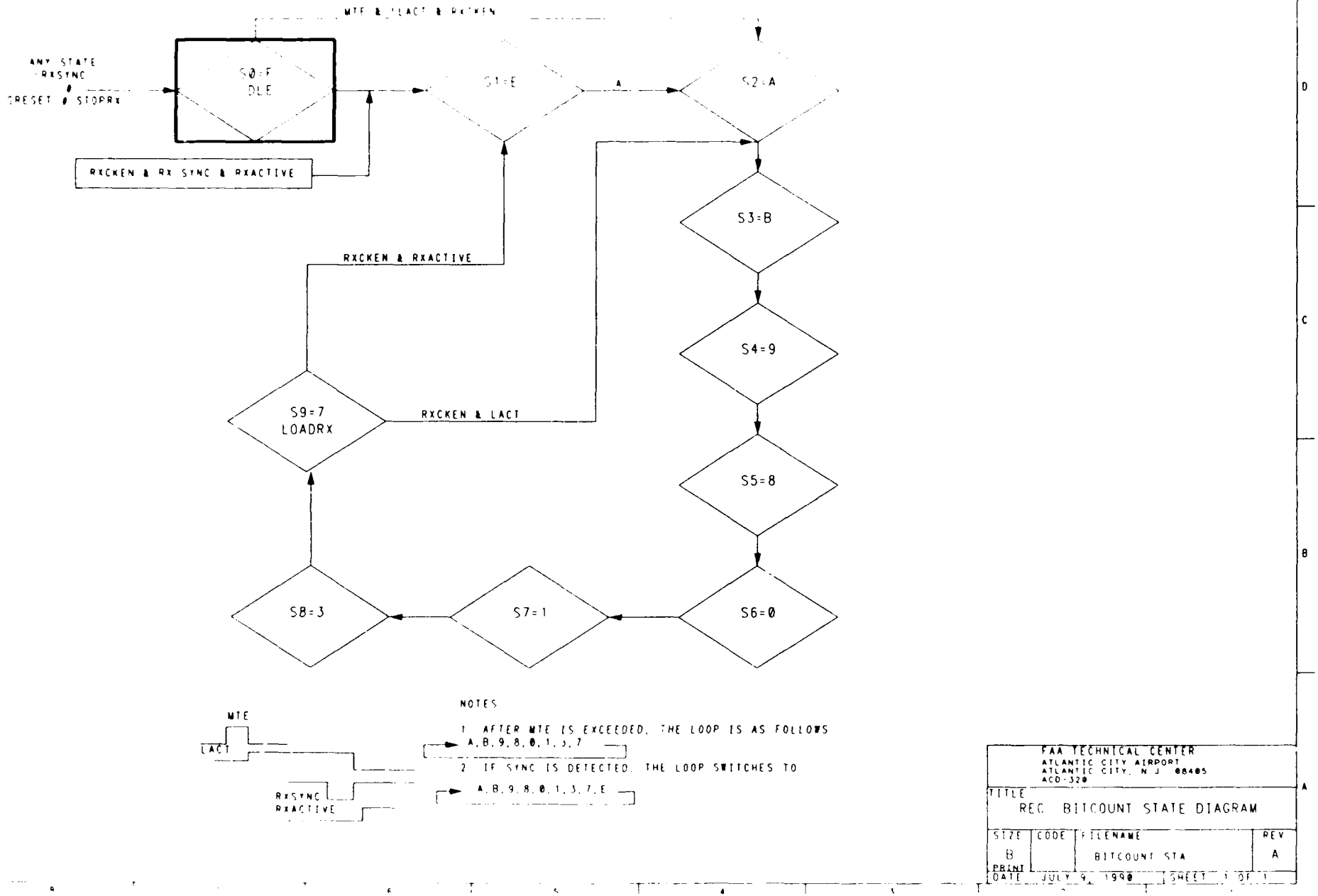


FIGURE 8. ATC BIT COUNTER STATE DIAGRAM

Block #	Data Address(start)	Status Address (start)
1	EF701080	EF701004
2	EF701100	EF701008
3	EF701180	EF70100C
4	EF701200	EF701010
5	EF701280	EF701014
6	EF701300	EF701018
7	EF701380	EF70101C

All data are actually stored in the receive data memory under the control of the status logic. The state diagram of this function is shown on figure 9. As indicated, this state diagram actually contains three separate loops. It is normally in state 0 (IDLE) awaiting a command to move to one of the other loops. It is preset to this state on **power up or reset** and is returned there upon leaving any of the other loops. The main control signal for the status logic is LDRX which comes from BITCOUNT.ABL during state 9 of its loop. The status logic first performs a **lead edge detection** of this signal since it lasts for the duration on one of the incoming receive data clocks. If no end of message (RXEOM) has been detected, the state controller goes to state 1. U47 then produces the write enable pulse, RXWR, to write the data into RAM. On the following clock, it goes to state 2 which results in En_Rmac which enables the address counter to increment to the next address. On the following clock, it returns to state 0 to await the next command. This loop will be the normal mode after the Monitor threshold has been exceeded until Sync has been detected.

When a receive sync pattern is detected, U42 produces **RXACTIVE**, which is sent to the status controller. It performs a "lead edge detection" of this signal which is used to send the state controller to state E. This is used to generate **Clr_Rmac** which is used to clear the memory address counter so that the data will be written over that stored previously after **MTE**. It remains in state E only for one clock period, after which it returns to state 0. Each nine bits thereafter, **LDRX** occurs and the loop is repeated to store all the incoming data in the Receive data memory. If **LDRX** is received after **RXEOM** and **RXACTIVE** have occurred, it signifies the end of the message and the status controller goes to the third loop to write the data and append the status data to the block. The sequence is as follows: At the lead edge of **LDRX** the state controller goes to state 3 which produces **RXWR** to write the last byte of data into the RAM. It then goes to state 4 which produces **Latlad** which is used to latch the address of the last byte into U51 and U52 (these data are to be included in the status block). On the following clock, it goes to state 5 which produces **Lo_Rmac** (which loads the memory address counter with the address for the status block for the particular group of data which has just been stored) and sets **STAT** to notify the logic that the Status portion of the receive cycle is in progress. **STAT** is used to disable the outputs of the input shift register U37 and to select the addresses of the status blocks in **RADCON.ABL** (the memory address counter

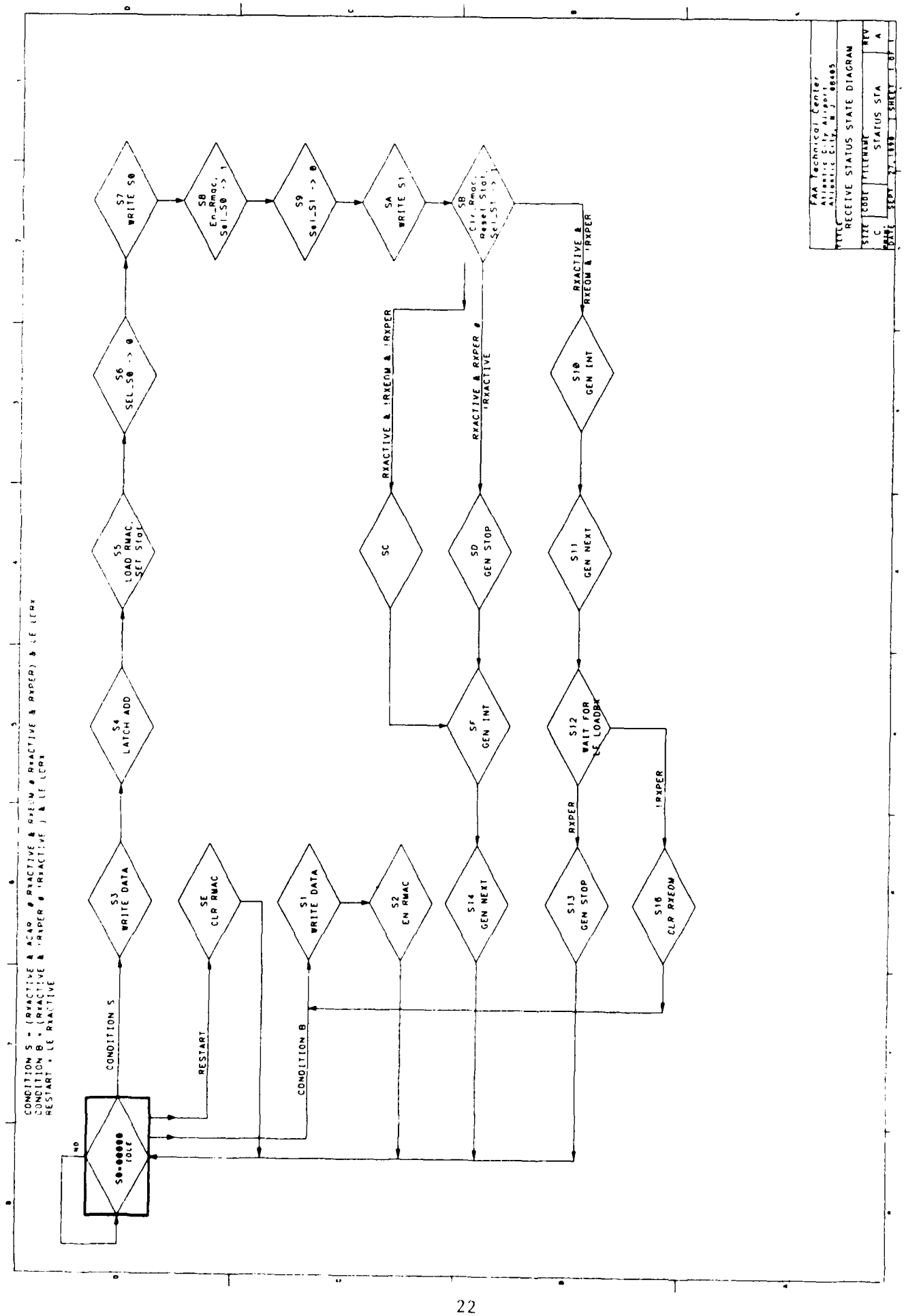


FIGURE 9. ATC RECEIVE STATUS STATE DIAGRAM

logic). The following clock takes the controller to state 6 which produces **Sel_S0** which is used to enable the contents of U52 to the Receive memory data bus (**RXDA<0-7>**). The contents of U52 were latched during state 4 and include the status of the following signals:

1. LACT - This signal is high from MTE to RXACTIVE and indicates to the processor that the data is a valid message (1) or input raw data (0).

2. ABORT - This notifies that processor that an ABORT message has been received. This consists of nine consecutive zeros.

3. HANDSHAKE BIT 7 - This bit is written to a zero by the processor each time it reads the status data and is written to a 1 each time a new status word is stored by the hardware. This bit is used to detect an out of sync condition between the hardware and software.

4. The two Msb's of the Last valid address.

The following clock takes the state controller to state 7 which produces **RXWR** to write the contents of U52 to memory. The following clock takes it to state 8 which produces **En_Rmac** to increment the address counter and reset **Sel_S0**. The following clock takes it to state 9 which produces **Sel_S1** to enable the contents of U51 to be placed on **RXDA<0-7>**. The following clock takes it to state A which produces another **RXWR** and writes the contents of U51 (the 8 Lsb's of the Last valid address) into memory. The following clock produces state B which produces **Clr_Rmac**, Resets **Stat** and sets **-Sel_S1** to a 1. On the following clock the logic takes one of three branches depending on the conditions. The normal exit loop for a valid message is shown in the sequence of appendix B, sheet B-9. This is for a message with a valid EOM and no parity error. If the **RXACTIVE** is a 1 and a parity error has occurred, the next state is SD, which generates **STOP** and then goes to SF on the following clock to generate the Receive interrupt. It then goes to S14 which generates **NEXT** to increment the address counter to the next block for the next message. The same loop is also followed if 128 bytes are stored after MTE without detecting a receive sync pattern. The only difference will be the fact that LACT will be a 1 to identify the block as entirely raw data after MTE was exceeded. If the state controller is at state SB with **RXACTIVE & !RXEOM & !RXPER** (the normal condition if a message exceeds 128 bytes), the state controller will go to state SC. From here it goes to S14 which generates **NEXT** to increment the address counter to the next block, but it skips the states which will generate an interrupt. An interrupt is generated only at the end of a sequence (a valid EOM or a parity error).

RECEIVE PARITY LOGIC.

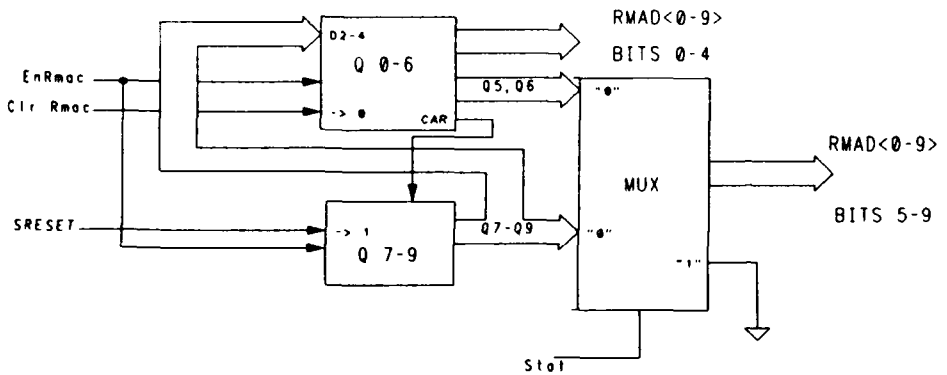
The parity of incoming data is checked by RXPAR.ABL (U38). It monitors the output of the Input data shift register U37, and the input parity bit (RXPARIN) to see if the received parity matches that computed from the input data. If the expected and received parity bits are different and LDRX is a 1, RXPER is generated to signify a parity error has occurred. If this occurs, it will result in the generation of RXEOM by EOMDET.ABL (U44) and storage of input data will be aborted.

EOM DETECTION.

The EOM detection logic monitors the output of the receive data input shift register. The normal EOM character is B1. This logic is disabled on the character following the detection of LRCPREP, as the value of this byte can be anything and must not be confused with the normal EOM. A receive message is also terminated whenever a parity error is detected.

RECEIVE MEMORY ADDRESS CONTROL.

This function is not a straight forward counter because of the way in which the data are organized in memory. The function is performed in the module called RADCON.ABL which is comprised of U49 and U50. A simplified block diagram of the function is shown on figure 10. The function consists of two separate counters and a multiplexor. A 7-bit counter addresses the Lsb's and a 3-bit counter accesses the Msb's of the memory. When STAT is not active (when not in the status loop), the outputs of the counters are selected via the multiplexor. If STAT is active, the four Lsb's are selected and the remaining six are forced to zero. The four Lsb's were preset to the proper address as part of the status loop so that the status block can be associated with the particular block of 128 bytes stored in RAM. Normally, the three Msb's identify the 128 byte block, but if a message exceeds 128 bytes, a CAR from the 7-bit counter increments the Msb counter to the next block and message storage continues. The overflow will be flagged by the value of the "last valid address" in the status area of the message. **The status message will be in the area associated with the last block of data.**



FAA TECHNICAL CENTER
ATLANTIC CITY AIRPORT
ATLANTIC CITY, N J 08405
ACD-320

TITLE: REPLY MEMORY ADDRESS CONTROL

SIZE	CODE	FILENAME	REV
8 PRINT	LJW	RADCON.DWG	A

DATE JULY 9, 1990 | SHEET 1 OF 1

FIGURE 10. ATC REPLY MEMORY ADDRESS CONTROL BLOCK DIAGRAM

APPENDIX A
TRANSMIT DATA SEQUENCES

APPENDIX A
TRANSMIT DATA OUTPUT SEQUENCES

This file is for the transmit logic operation. The message selected is the standard 8 byte message (0,1,2,3,4,5,6,B1).

THIS DATA SET IS A TRANSMIT CYCLE WITH TRANSMIT LOGIC ENABLED.
The logic analyzer is set up to store only if TXCKEN=1.
It is set to trigger on TXACTIVE.

When the transmit logic is ENABLED, the loop transmit control loop is as follows: (0,1,3,2,6,7,5,4,C,D,F,E,A,B,9,8,10,11,13,12,16,17,15,14,1C,1D,1F,1E,12,16,17,15,14 etc.) In this mode, the loop is 9 states long after the initial sequence.

SEQ		TXCKEN	STRT	TXACTIVE	TXCONTROL	ENMADEM	ENMTXMEM	ENMTCTR	LDTINT	TXDA<07>	ADD	TX01'S	TX1'S	LOADSR	TXADDAT	
505		1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	
506		1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	
507		1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	
508		1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	
509		1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	
510	T	1	1	1	00	0	0	0	1	0	00	008	0	1	0	0	TXACTIVE occurs
511		1	1	1	01	0	0	0	1	0	00	008	1	0	0	1	TX10'S->0,TX0'S->0,TXCONT->01
512		1	1	1	03	0	0	0	0	0	00	000	1	0	0	0	BYTE COUNTER LOADED
513		1	1	1	02	0	0	0	1	0	00	000	1	0	0	0	
514		1	1	1	06	0	0	0	1	0	00	000	1	0	0	0	
515		1	1	1	07	0	0	0	1	0	00	000	1	0	0	0	
516		1	1	1	05	0	0	0	1	0	00	000	1	0	0	0	
517		1	1	1	04	0	0	0	1	0	00	000	1	0	0	0	
518		1	1	1	0C	0	0	0	1	0	00	000	1	0	0	0	
519		1	1	1	0D	0	0	0	1	0	00	000	1	0	0	0	
520		1	1	1	0F	0	0	0	1	0	00	000	1	0	0	0	
521		1	1	1	0E	0	0	0	1	0	00	000	1	0	0	0	
522		1	1	1	0A	0	0	0	1	0	00	000	1	0	0	0	
523		1	1	1	0B	0	0	0	1	0	00	000	1	0	0	0	
524		1	1	1	09	0	0	0	1	0	00	000	1	0	0	0	
525		1	1	1	08	0	0	0	1	0	00	000	1	0	0	0	
526		1	1	1	10	0	0	0	1	0	00	000	1	0	0	0	
527		1	1	1	11	0	0	0	1	0	00	000	1	0	0	0	
528		1	1	1	13	0	0	0	1	0	00	000	1	1	0	0	TX10'S->1(GEN SYNC=1)
529		1	1	1	12	0	0	0	1	0	00	000	0	0	1	1	LOADSR -> 00
530		1	1	1	16	1	0	0	1	0	00	000	0	0	0	1	En_Memad generated
531		1	1	1	17	0	0	1	1	0	01	001	0	0	0	0	TXMAD->01
532		1	1	1	15	0	0	0	1	0	01	001	0	0	0	0	
533		1	1	1	14	0	0	0	1	0	01	001	0	0	0	0	
534		1	1	1	1C	0	0	0	1	0	01	001	0	0	0	0	
535		1	1	1	1D	0	0	0	1	0	01	001	0	0	0	0	

536	1	1	1	1F	0	0	0	1	0	01	001	0	0	0	0	
SEQ	T	S	T	TX	E	E	E	L	T	TX	ADD	T	T	L	T	
	X	T	X	C	N	N	N	D	X	DA		X	X	O	X	
	C	A	A	O	M	T	T	T	I	<		0	1	A	D	
	K	R	C	N	E	X	C	C	N	0		'	0	D	A	
	E	T	T	T	M	M	T	T	T	-		S	'	S	T	
	N		I	R	A	E				7			S	R	A	
			V	O	D	M				>						
			E	L												
537	1	1	1	1E	0	0	0	1	0	01	001	0	0	0	0	
538	1	1	1	12	0	0	0	1	0	01	001	0	0	1	0	LOADSR -> 01
539	1	1	1	16	1	0	0	1	0	01	001	0	0	0	0	En_Memad generated
540	1	1	1	17	0	0	1	1	0	02	002	0	0	0	0	TXMAD->02
541	1	1	1	15	0	0	0	1	0	02	002	0	0	0	0	
542	1	1	1	14	0	0	0	1	0	02	002	0	0	0	0	
543	1	1	1	1C	0	0	0	1	0	02	002	0	0	0	0	
544	1	1	1	1D	0	0	0	1	0	02	002	0	0	0	0	
545	1	1	1	1F	0	0	0	1	0	02	002	0	0	0	0	
546	1	1	1	1E	0	0	0	1	0	02	002	0	0	0	0	
547	1	1	1	12	0	0	0	1	0	02	002	0	0	1	1	LOADSR -> 02
548	1	1	1	16	1	0	0	1	0	02	002	0	0	0	0	En_Memad generated
549	1	1	1	17	0	0	1	1	0	03	003	0	0	0	0	TXMAD->03
550	1	1	1	15	0	0	0	1	0	03	003	0	0	0	0	
551	1	1	1	14	0	0	0	1	0	03	003	0	0	0	0	
552	1	1	1	1C	0	0	0	1	0	03	003	0	0	0	0	
553	1	1	1	1D	0	0	0	1	0	03	003	0	0	0	0	
554	1	1	1	1F	0	0	0	1	0	03	003	0	0	0	0	
555	1	1	1	1E	0	0	0	1	0	03	003	0	0	0	1	
556	1	1	1	12	0	0	0	1	0	03	003	0	0	1	0	LOADSR -> 03
557	1	1	1	16	1	0	0	1	0	03	003	0	0	0	1	En_Memad generated
558	1	1	1	17	0	0	1	1	0	04	004	0	0	0	0	TXMAD->04
559	1	1	1	15	0	0	0	1	0	04	004	0	0	0	0	
560	1	1	1	14	0	0	0	1	0	04	004	0	0	0	0	
561	1	1	1	1C	0	0	0	1	0	04	004	0	0	0	0	
562	1	1	1	1D	0	0	0	1	0	04	004	0	0	0	0	
563	1	1	1	1F	0	0	0	1	0	04	004	0	0	0	0	
564	1	1	1	1E	0	0	0	1	0	04	004	0	0	0	1	
565	1	1	1	12	0	0	0	1	0	04	004	0	0	1	1	LOADSR -> 04
566	1	1	1	16	1	0	0	1	0	04	004	0	0	0	0	En_Memad generated
567	1	1	1	17	0	0	1	1	0	05	005	0	0	0	0	TXMAD->05
568	1	1	1	15	0	0	0	1	0	05	005	0	0	0	0	
569	1	1	1	14	0	0	0	1	0	05	005	0	0	0	0	
570	1	1	1	1C	0	0	0	1	0	05	005	0	0	0	0	
571	1	1	1	1D	0	0	0	1	0	05	005	0	0	0	0	
572	1	1	1	1F	0	0	0	1	0	05	005	0	0	0	1	
573	1	1	1	1E	0	0	0	1	0	05	005	0	0	0	0	
574	1	1	1	12	0	0	0	1	0	05	005	0	0	1	0	LOADSR -> 05
575	1	1	1	16	1	0	0	1	0	05	005	0	0	0	1	En_Memad generated
576	1	1	1	17	0	0	1	1	0	06	006	0	0	0	0	TXMAD->06
577	1	1	1	15	0	0	0	1	0	06	006	0	0	0	0	
578	1	1	1	14	0	0	0	1	0	06	006	0	0	0	0	
579	1	1	1	1C	0	0	0	1	0	06	006	0	0	0	0	
580	1	1	1	1D	0	0	0	1	0	06	006	0	0	0	0	

581	1	1	1	1F	0	0	0	1	0	06	006	0	0	0	1	
SEQ	T	S	T	TX	E	E	E	L	T	TX	ADD	T	T	L	T	
	X	T	X	C	N	N	N	D	X	DA		X	X	O	X	
	C	A	A	O	M	T	T	T	I	<		0	1	A	D	
	K	R	C	N	E	X	C	C	N	0		'	0	D	A	
	E	T	T	T	M	M	T	T	T	-		S	'	S	T	
	N		I	R	A	E				7			S	R	A	
			V	O	D	M				>						
			E	L												
582	1	1	1	1E	0	0	0	1	0	06	006	0	0	0	0	
583	1	1	1	12	0	0	0	1	0	06	006	0	0	1	1	LOADSR -> 06
584	1	1	1	16	1	0	0	1	0	06	006	0	0	0	1	En_Memad generated
585	1	1	1	17	0	0	1	1	0	B1	007	0	0	0	0	TXMAD->07
586	1	1	1	15	0	0	0	1	0	B1	007	0	0	0	0	
587	1	1	1	14	0	0	0	1	0	B1	007	0	0	0	0	
588	1	1	1	1C	0	0	0	1	0	B1	007	0	0	0	0	
589	1	1	1	1D	0	0	0	1	0	B1	007	0	0	0	0	
590	1	1	1	1F	0	0	0	1	0	B1	007	0	0	0	1	
591	1	1	1	1E	0	0	0	1	0	B1	007	0	0	0	1	
592	1	1	1	12	0	0	0	1	0	B1	007	0	0	1	0	LOADSR -> B1
593	1	1	1	16	1	0	0	1	0	B1	007	0	0	0	1	En_Memad generated
594	1	1	1	17	0	0	1	1	0	00	008	0	0	0	1	TXMAD->08
595	1	1	1	15	0	0	0	1	0	00	008	0	0	0	0	
596	1	1	1	14	0	0	0	1	0	00	008	0	0	0	1	
597	1	1	1	1C	0	0	0	1	0	00	008	0	0	0	1	
598	1	1	1	1D	0	0	0	1	0	00	008	0	0	0	0	
599	1	1	1	1F	0	0	0	1	0	00	008	0	0	0	0	
600	1	1	1	1E	0	0	0	1	0	00	008	0	0	0	0	
601	1	1	1	12	0	0	0	1	0	00	008	0	1	0	1	
602	1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	
603	1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	
604	1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	

THIS DATA SET IS A TRANSMIT CYCLE WITH TRANSMIT LOGIC DISABLED.
The logic analyzer is set up to store only if TXCKEN=1.
It is set to trigger on TXACTIVE.

When the transmit logic is DISABLED, the loop transmit control loop is as follows: (0,1,3,2,6,7,5,4,C,D,F,E,A,B,9,8,10,11,13,12,16,17,15,14,1C,1D,1F,12,16,17,15,14 etc.) In this mode, the loop is 8 states long after the initial sequence.

SEQ		TXCKEN	STARTR	TXACTIVE	TXCONTROL	ENMEMAD	ENTXCMEM	ENTCTCTN	LDTCTN	TXIDLE	TXDA<07>	ADDR	TXS	TX1'S	LOADSR	TXDATA	
502		1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	
503		1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	
504		1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	
505		1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	
506		1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	
507		1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	
508		1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	
509		1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	
510	T	1	1	1	00	0	0	0	1	0	00	008	0	1	0	0	TXACTIVE occurs
511		1	1	1	01	0	0	0	1	0	00	008	1	0	0	1	TX10'S->0,TX0'S->0,TXCONT->01
512		1	1	1	03	0	0	0	0	0	00	000	1	0	0	0	
513		1	1	1	02	0	0	0	1	0	00	000	1	0	0	0	
514		1	1	1	06	0	0	0	1	0	00	000	1	0	0	0	
515		1	1	1	07	0	0	0	1	0	00	000	1	0	0	0	
516		1	1	1	05	0	0	0	1	0	00	000	1	0	0	0	
517		1	1	1	04	0	0	0	1	0	00	000	1	0	0	0	
518		1	1	1	0C	0	0	0	1	0	00	000	1	0	0	0	
519		1	1	1	0D	0	0	0	1	0	00	000	1	0	0	0	
520		1	1	1	0F	0	0	0	1	0	00	000	1	0	0	0	
521		1	1	1	0E	0	0	0	1	0	00	000	1	0	0	0	
522		1	1	1	0A	0	0	0	1	0	00	000	1	0	0	0	
523		1	1	1	0B	0	0	0	1	0	00	000	1	0	0	0	
524		1	1	1	09	0	0	0	1	0	00	000	1	0	0	0	
525		1	1	1	08	0	0	0	1	0	00	000	1	0	0	0	
526		1	1	1	10	0	0	0	1	0	00	000	1	0	0	0	
527		1	1	1	11	0	0	0	1	0	00	000	1	0	0	0	
528		1	1	1	13	0	0	0	1	0	00	000	1	1	0	0	
529		1	1	1	12	0	0	0	1	0	00	000	0	0	1	0	LOADSR ->00
530		1	1	1	16	1	0	0	1	0	00	000	0	0	0	0	En_Memad generated
531		1	1	1	17	0	0	1	1	0	01	001	0	0	0	0	
532		1	1	1	15	0	0	0	1	0	01	001	0	0	0	0	
533		1	1	1	14	0	0	0	1	0	01	001	0	0	0	0	
534		1	1	1	1C	0	0	0	1	0	01	001	0	0	0	0	
535		1	1	1	1D	0	0	0	1	0	01	001	0	0	0	0	
536		1	1	1	1F	0	0	0	1	0	01	001	0	0		0	
537		1	1	1	12	0	0	0	1	0	01	001	0	0	1	0	LOADSR ->01
538		1	1	1	16	1	0	0	1	0	01	001	0	0	0	0	En_Memad generated

SEQ	TXCKEN	STAR T	TXACTIVE	TXCONTROL	ENMADEM	ENMTXEM	ENTCT	LDTCT	TXIN T	TXDA<0-7>	ADD	TX0'S	TX1'S	LOADSR	TXDATA	
539	1	1	1	17	0	0	1	1	0	02	002	0	0	0	0	
540	1	1	1	15	0	0	0	1	0	02	002	0	0	0	0	
541	1	1	1	14	0	0	0	1	0	02	002	0	0	0	0	
542	1	1	1	1C	0	0	0	1	0	02	002	0	0	0	0	
543	1	1	1	1D	0	0	0	1	0	02	002	0	0	0	0	
544	1	1	1	1F	0	0	0	1	0	02	002	0	0	0	0	
545	1	1	1	12	0	0	0	1	0	02	002	0	0	1	0	LOADSR ->02
546	1	1	1	16	1	0	0	1	0	02	002	0	0	0	1	En_Memad generated
547	1	1	1	17	0	0	1	1	0	03	003	0	0	0	0	
548	1	1	1	15	0	0	0	1	0	03	003	0	0	0	0	
549	1	1	1	14	0	0	0	1	0	03	003	0	0	0	0	
550	1	1	1	1C	0	0	0	1	0	03	003	0	0	0	0	
551	1	1	1	1D	0	0	0	1	0	03	003	0	0	0	0	
552	1	1	1	1F	0	0	0	1	0	03	003	0	0	0	0	
553	1	1	1	12	0	0	0	1	0	03	003	0	0	1	1	LOADSR ->03
554	1	1	1	16	1	0	0	1	0	03	003	0	0	0	0	En_Memad generated
555	1	1	1	17	0	0	1	1	0	04	004	0	0	0	0	
556	1	1	1	15	0	0	0	1	0	04	004	0	0	0	0	
557	1	1	1	14	0	0	0	1	0	04	004	0	0	0	0	
558	1	1	1	1C	0	0	0	1	0	04	004	0	0	0	0	
559	1	1	1	1D	0	0	0	1	0	04	004	0	0	0	0	
560	1	1	1	1F	0	0	0	1	0	04	004	0	0	0	0	
561	1	1	1	12	0	0	0	1	0	04	004	0	0	1	1	LOADSR ->04
562	1	1	1	16	1	0	0	1	0	04	004	0	0	0	1	En_Memad generated
563	1	1	1	17	0	0	1	1	0	05	005	0	0	0	0	
564	1	1	1	15	0	0	0	1	0	05	005	0	0	0	0	
565	1	1	1	14	0	0	0	1	0	05	005	0	0	0	0	
566	1	1	1	1C	0	0	0	1	0	05	005	0	0	0	0	
567	1	1	1	1D	0	0	0	1	0	05	005	0	0	0	0	
568	1	1	1	1F	0	0	0	1	0	05	005	0	0	0	1	
569	1	1	1	12	0	0	0	1	0	05	005	0	0	1	0	LOADSR ->05
570	1	1	1	16	1	0	0	1	0	05	005	0	0	0	0	En_Memad generated
571	1	1	1	17	0	0	1	1	0	06	006	0	0	0	0	
572	1	1	1	15	0	0	0	1	0	06	006	0	0	0	0	
573	1	1	1	14	0	0	0	1	0	06	006	0	0	0	0	
574	1	1	1	1C	0	0	0	1	0	06	006	0	0	0	0	
575	1	1	1	1D	0	0	0	1	0	06	006	0	0	0	0	
576	1	1	1	1F	0	0	0	1	0	06	006	0	0	0	1	
577	1	1	1	12	0	0	0	1	0	06	006	0	0	1	0	LOADSR ->06
578	1	1	1	16	1	0	0	1	0	06	006	0	0	0	1	En_Memad generated
579	1	1	1	17	0	0	1	1	0	B1	007	0	0	0	0	
580	1	1	1	15	0	0	0	1	0	B1	007	0	0	0	0	
581	1	1	1	14	0	0	0	1	0	B1	007	0	0	0	0	
582	1	1	1	1C	0	0	0	1	0	B1	007	0	0	0	0	
583	1	1	1	1D	0	0	0	1	0	B1	007	0	0	0	0	

SEQ	TXCKEN	STAR T	TXACTIVE	TXCONTROL	ENMEMAD	ENMEXMEM	ENTCT	LDTCT	TXINT	TXDA<07>	ADD	TX0'S	TX1'S	LOADSR	TXDATA	
584	1	1	1	1F	0	0	0	1	0	B1	007	0	0	0	1	
585	1	1	1	12	0	0	0	1	0	B1	007	0	0	1	1	LOADSR ->B1
586	1	1	1	16	1	0	0	1	0	B1	007	0	0	0	0	En_Memad generated
587	1	1	1	17	0	0	1	1	0	00	008	0	0	0	1	
588	1	1	1	15	0	0	0	1	0	00	008	0	0	0	0	
589	1	1	1	14	0	0	0	1	0	00	008	0	0	0	1	
590	1	1	1	1C	0	0	0	1	0	00	008	0	0	0	1	
591	1	1	1	1D	0	0	0	1	0	00	008	0	0	0	0	
592	1	1	1	1F	0	0	0	1	0	00	008	0	0	0	0	
593	1	1	1	12	0	0	0	1	0	00	008	0	1	0	0	
594	1	1	0	00	0	1	0	1	0	01	008	0	1	1	1	TXCON->0 (CYCLE COMPLETE)
595	1	1	0	00	0	1	0	1	0	01	008	0	1	1	0	

APPENDIX B
RECEIVE DATA SEQUENCES

APPENDIX B
RECEIVE DATA INPUT SEQUENCES

This file is collected to show the Monitor Threshold Logic operation (MTE). The input for all of these files is the standard 8 byte message ending in a"b1". The data is "0,1,2,3,4,5,6,b1". For this file, Receive Sync detection was disabled, the MTE threshold was "6" and the transmit logic was "enabled".

The output data in the loopback mode was 00180004050302416Odb15555555 to add 127.

The status data was "d1 7f 00", indicating the lack of a detected sync, a full 128 bytes of data in buffer No.1.

The logic analyzer was set to store data only if RXCK=1 OR STA=01.

SEQ		RXCK	DATACT	RXACTIVE	SRXSTYNC	BITCOUNT	ACTIVE	STATUS	LACT	RXPER	RXD<0-7>	ADD	MTE	EN	COMMENTS
493		1	1	0	11	F	0	00	0	1	AA	200	0	1	
494		1	0	0	11	F	0	00	0	1	55	200	0	1	
495		1	1	0	11	F	0	00	0	1	AA	200	0	1	
496		1	0	0	11	F	0	00	0	1	55	200	0	1	
497		1	1	0	11	F	0	00	0	1	AA	200	0	1	
498		1	0	0	11	F	0	00	0	1	55	200	0	1	
499		1	1	0	11	F	0	00	0	1	AA	200	0	1	
500		1	0	0	11	F	0	00	0	1	55	200	0	1	
501		1	1	0	11	F	0	00	0	1	AA	200	0	1	
502		1	0	0	11	F	0	00	0	1	55	200	0	1	
503		1	1	0	11	F	0	00	0	1	AA	200	0	1	
504		1	0	0	11	F	0	00	0	1	55	200	0	1	FIRST 0 OCCURS FOR SYNC
505		1	0	0	11	F	0	00	0	1	AA	200	0	1	
506		1	0	0	11	F	0	00	0	1	54	200	0	1	
507		1	0	0	11	F	0	00	0	1	A8	200	0	1	
508		1	0	0	11	F	0	00	0	1	50	200	0	1	
509		1	0	0	11	F	0	00	0	1	A0	200	0	1	
510	T	1	0	0	11	F	0	00	0	1	40	200	1	0	
511		1	0	0	11	A	0	00	1	1	80	200	0	0	MTE OCCURS (7 ZEROS) - BIT COUNTER -> A
512		1	0	0	11	B	0	00	1	1	00	200	0	0	BIT COUNTER IN 8 STATE LOOP
513		1	0	0	11	9	0	00	1	1	00	200	0	0	(A,B,9,8,0,1,3,7,A ETC.)
514		1	0	0	11	8	0	00	1	1	00	200	0	0	
515		1	0	0	11	0	0	00	1	1	00	200	0	0	
516		1	0	0	11	1	0	00	1	1	00	200	0	0	
517		1	0	0	11	3	0	00	1	1	00	200	0	0	
518		0	0	0	11	7	0	01	1	1	00	200	0	0	1ST BYTE (00) WRITTEN TO ADD=200
519		1	0	0	11	7	0	00	1	1	00	201	0	0	
520		1	0	0	11	A	0	00	1	1	00	201	0	0	
521		1	0	0	11	B	0	00	1	1	00	201	0	0	
522		1	1	0	11	9	0	00	1	1	00	201	0	0	SYNC 1 OCCURS (SYNC DISABLED)

SEQ	RXCK	DATA	RXACTIVE	SRIXSTYNC	BITCOUNT	ACTIVE	STATUS	LACT	RXPER	RXDA<0-7>	ADD	MTE	EN	COMMENTS
523	1	1	0	11	8	0	00	1	1	01	201	0	1	PARITY 1ST BYTE
524	1	0	0	11	0	0	00	1	1	03	201	0	1	
525	1	0	0	11	1	0	00	1	1	06	201	0	1	
526	1	0	0	11	3	0	00	1	1	0C	201	0	1	
527	0	0	0	11	7	0	01	1	1	18	201	0	1	2ND BYTE (18) WRITTEN TO ADD=201
528	1	0	0	11	7	0	00	1	1	18	202	0	1	
529	1	0	0	11	A	0	00	1	1	30	202	0	1	
530	1	0	0	11	B	0	00	1	1	60	202	0	1	
531	1	0	0	11	9	0	00	1	1	C0	202	1	0	
532	1	0	0	11	8	0	00	1	1	80	202	0	0	
533	1	0	0	11	0	0	00	1	1	00	202	0	0	
534	1	0	0	11	1	0	00	1	1	00	202	0	0	
535	1	0	0	11	3	0	00	1	1	00	202	0	0	
536	0	0	0	11	7	0	01	1	1	00	202	0	0	3RD BYTE (00) WRITTEN TO ADD=202
537	1	0	0	11	7	0	00	1	1	00	203	0	0	
538	1	0	0	11	A	0	00	1	1	00	203	0	0	
539	1	0	0	11	B	0	00	1	1	00	203	0	0	
540	1	0	0	11	9	0	00	1	1	00	203	0	0	
541	1	0	0	11	8	0	00	1	1	00	203	0	0	
542	1	1	0	11	0	0	00	1	1	00	203	0	0	
543	1	0	0	11	1	0	00	1	1	01	203	0	1	
544	1	0	0	11	3	0	00	1	1	02	203	0	1	
545	0	0	0	11	7	0	01	1	1	04	203	0	1	4TH BYTE (04) WRITTEN TO ADD=203
546	1	0	0	11	7	0	00	1	1	00	204	0	1	
547	1	0	0	11	A	0	00	1	1	08	204	0	1	
548	1	0	0	11	B	0	00	1	1	10	204	0	1	
549	1	0	0	11	9	0	00	1	1	20	204	0	1	
550	1	0	0	11	8	0	00	1	1	40	204	1	0	
551	1	1	0	11	0	0	00	1	1	80	204	0	0	
552	1	0	0	11	1	0	00	1	1	01	204	0	1	
553	1	1	0	11	3	0	00	1	1	02	204	0	1	
554	0	1	0	11	7	0	01	1	1	00	204	0	1	5TH BYTE (00) WRITTEN TO ADD=204
555	1	0	0	11	7	0	00	1	1	05	205	0	1	
556	1	0	0	11	A	0	00	1	1	0A	205	0	1	
557	1	0	0	11	B	0	00	1	1	14	205	0	1	
558	1	0	0	11	9	0	00	1	1	28	205	0	1	
559	1	0	0	11	8	0	00	1	1	50	205	0	1	
560	1	0	0	11	0	0	00	1	1	A0	205	0	1	
561	1	1	0	11	1	0	00	1	1	40	205	1	0	
562	1	1	0	11	3	0	00	1	1	81	205	0	1	
563	0	1	0	11	7	0	01	1	1	03	205	0	1	6TH BYTE (03) WRITTEN TO ADD=205
564	1	0	0	11	7	0	00	1	1	03	206	0	1	
565	1	0	0	11	A	0	00	1	1	06	206	0	1	
566	1	0	0	11	B	0	00	1	1	0C	206	0	1	
567	1	0	0	11	9	0	00	1	1	18	206	0	1	

SEQ	RXCK	DATA	RXACTIVE	SRIXSTYNC	BITCOUNT	ACTIVE	STATUS	LACT	RXPER	RXD A<0-7>	ADD	MTE	EN	COMMENTS
568	1	0	0	11	8	0	00	1	1	30	206	0	1	
569	1	0	0	11	0	0	00	1	1	60	206	0	1	
570	1	1	0	11	1	0	00	1	1	C0	206	1	0	
571	1	0	0	11	3	0	00	1	1	81	206	0	1	
572	0	0	0	11	7	0	01	1	1	02	206	0	1	7TH BYTE (02) WRITTEN TO ADD=206
573	1	0	0	11	7	0	00	1	1	02	207	0	1	
574	1	1	0	11	A	0	00	1	1	04	207	0	1	
575	1	0	0	11	B	0	00	1	1	09	207	0	1	
576	1	0	0	11	9	0	00	1	1	12	207	0	1	
577	1	0	0	11	8	0	00	1	1	24	207	0	1	
578	1	0	0	11	0	0	00	1	1	48	207	0	1	
579	1	0	0	11	1	0	00	1	1	90	207	0	1	
580	1	1	0	11	3	0	00	1	1	20	207	0	1	
581	0	1	0	11	7	0	01	1	1	41	207	0	1	8TH BYTE (41) WRITTEN TO ADD=207
582	1	0	0	11	7	0	00	1	1	41	208	0	1	
583	1	1	0	11	A	0	00	1	1	82	208	0	1	
584	1	1	0	11	B	0	00	1	1	05	208	0	1	
585	1	0	0	11	9	0	00	1	1	08	208	0	1	
586	1	0	0	11	8	0	00	1	1	16	208	0	1	
587	1	0	0	11	0	0	00	1	1	2C	208	0	1	
588	1	0	0	11	1	0	00	1	1	58	208	0	1	
589	1	0	0	11	3	0	00	1	1	B0	208	0	1	
590	0	0	0	11	7	0	01	1	1	60	208	0	1	9TH BYTE (60) WRITTEN TO ADD=208
591	1	1	0	11	7	0	00	1	1	60	209	0	1	
592	1	1	0	11	A	0	00	1	1	C1	209	0	1	
593	1	0	0	11	B	0	00	1	1	83	209	0	1	
594	1	1	0	11	9	0	00	1	1	06	209	0	1	PARITY BIT OF LAST BYTE OF MESSAGE
595	1	1	0	11	8	0	00	1	1	0D	209	0	1	1ST BIT OF LAST BYTE (B1)
596	1	0	0	11	0	0	00	1	1	1B	209	0	1	
597	1	1	0	11	1	0	00	1	1	36	209	0	1	
598	1	1	0	11	3	0	00	1	1	6D	209	0	1	
599	0	1	0	11	7	0	01	1	1	DB	209	0	1	10TH BYTE (DB) WRITTEN TO ADD=209
600	1	0	0	11	7	0	00	1	1	DB	20A	0	1	
601	1	0	0	11	A	0	00	1	1	B6	20A	0	1	
602	1	0	0	11	B	0	00	1	1	6C	20A	0	1	
603	1	1	0	11	9	0	00	1	1	D8	20A	0	1	LAST BIT OF MESSAGE
604	1	0	0	11	8	0	00	1	1	B1	20A	0	1	
605	1	1	0	11	0	0	00	1	1	62	20A	0	1	
606	1	0	0	11	1	0	00	1	1	C5	20A	0	1	
607	1	1	0	11	3	0	00	1	1	8A	20A	0	1	
608	0	1	0	11	7	0	01	1	1	15	20A	0	1	11TH BYTE (15) WRITTEN TO ADD=20A
609	1	0	0	11	7	0	00	1	1	15	20B	0	1	
610	1	1	0	11	A	0	00	1	1	2A	20B	0	1	
611	1	0	0	11	B	0	00	1	1	55	20B	0	1	
612	1	1	0	11	9	0	00	1	1	AA	20B	0	1	

SEQ	RXCK	DATA	RXACTIVE	SRIXSTYNC	BITCOUNT	ACTIVE	STATUS	LACT	RXPER	RXDA<0-7>	ADD	MTE	EN	COMMENTS
613	1	0	0	11	8	0	00	1	1	55	20B	0	1	
614	1	1	0	11	0	0	00	1	1	AA	20B	0	1	
615	1	0	0	11	1	0	00	1	1	55	20B	0	1	
616	1	1	0	11	3	0	00	1	1	AA	20B	0	1	
617	0	1	0	11	7	0	01	1	1	55	20B	0	1	12TH BYTE (55) WRITTEN TO ADD=20B
618	1	0	0	11	7	0	00	1	1	55	20C	0	1	
619	1	1	0	11	A	0	00	1	1	AA	20C	0	1	
620	1	0	0	11	B	0	00	1	1	55	20C	0	1	
621	1	1	0	11	9	0	00	1	1	AA	20C	0	1	
622	1	0	0	11	8	0	00	1	1	55	20C	0	1	
623	1	1	0	11	0	0	00	1	1	AA	20C	0	1	
624	1	0	0	11	1	0	00	1	1	55	20C	0	1	
625	1	1	0	11	3	0	00	1	1	AA	20C	0	1	
626	0	1	0	11	7	0	01	1	1	55	20C	0	1	13TH BYTE (55) WRITTEN TO ADD=20C

This file is collected to show the Monitor Threshold Logic operation (MTE). The input for all of these files is the standard 8 byte message ending in a "b1". The data is "0,1,2,3,4,5,6,b1". For this file, Receive Sync detection was ENABLED, the MTE threshold was "6" and the transmit logic was "enabled".

The output data in the loopback mode was 00010203040506B1.

The status data was "82 87 00", indicating the good parity and 8 bytes of data in buffer No. 5.

The logic analyzer was set to store data only if RXCK=1 OR STA=01.

SEQ		RXCK	DATA	RXSYNDET	SRIXXSTYNC	BITCOUNT	RXACTIVE	STATUS	LACT	RXPER	RXDA<0-7>	ADD	MTE	EN	COMMENTS
497		1	1	1	11	F	0	00	0	1	AA	280	0	1	
498		1	0	0	11	F	0	00	0	1	55	280	0	1	
499		1	1	1	11	F	0	00	0	1	AA	280	0	1	
500		1	0	0	11	F	0	00	0	1	55	280	0	1	
501		1	1	1	11	F	0	00	0	1	AA	280	0	1	
502		1	0	0	11	F	0	00	0	1	55	280	0	1	
503		1	1	1	11	F	0	00	0	1	AA	280	0	1	
504		1	0	0	11	F	0	00	0	1	55	280	0	1	FIRST 0 OCCURS FOR SYNC
505		1	0	1	11	F	0	00	0	1	AA	280	0	1	
506		1	0	3	11	F	0	00	0	1	54	280	0	1	
507		1	0	2	11	F	0	00	0	1	A8	280	0	1	
508		1	0	6	11	F	0	00	0	1	50	280	0	1	
509		1	0	7	11	F	0	00	0	1	A0	280	0	1	
510	T	1	0	5	11	F	0	00	0	1	40	280	1	0	
511		1	0	4	11	A	0	00	1	1	80	280	0	0	MTE OCCURS(7 ZEROS)-BIT COUNTER->A
512		1	0	C	11	B	0	00	1	1	00	280	0	0	BIT COUNTER IN 8 STATE LOOP
513		1	0	D	11	9	0	00	1	1	00	280	0	0	(A,B,9,8,0,1,3,7,A etc.)
514		1	0	F	11	8	0	00	1	1	00	280	0	0	
515		1	0	E	11	0	0	00	1	1	00	280	0	0	
516		1	0	A	11	1	0	00	1	1	00	280	0	0	
517		1	0	B	11	3	0	00	1	1	00	280	0	0	
518		0	0	9	11	7	0	01	1	1	00	280	0	0	1st byte (00) written to ADD=280
519		1	0	9	11	7	0	00	1	1	00	281	0	0	
520		1	0	8	11	A	0	00	1	1	00	281	0	0	
521		1	0	8	01	B	0	00	1	1	00	281	0	0	SIXT OCCURS
522		1	1	8	00	F	0	00	0	0	00	281	0	0	RXSYNC OCCURS-LACT,RXPER RESET
523		1	1	0	11	F	1	00	0	0	01	280	0	0	RXACTIVE SET,ADD COUNT -> ADD=280
524		1	0	0	11	E	1	00	0	0	03	280	0	0	BIT COUNTER TO 9 STATE LOOP
525		1	0	1	11	A	1	00	0	0	06	280	0	0	(E,A,B,9,8,0,1,3,7,E,A etc.)
526		1	0	3	11	B	1	00	0	0	0C	280	0	0	
527		1	0	2	11	9	1	00	0	0	18	280	0	0	
528		1	0	6	11	8	1	00	•0	0	30	280	0	0	
529		1	0	7	11	0	1	00	0	0	60	280	0	0	

SEQ	RXCK	DATA	RXSYNDET	SRIXXSTYNC	BITCOUNTVE	RXACTIVE	STATUS	LACT	RXPER	RXD A<0-7>	ADD	MTE	EN	COMMENTS
530	1	0	5	11	1	1	00	0	0	C0	280	0	0	
531	1	0	4	11	3	1	00	0	0	80	280	0	0	
532	0	0	C	11	7	1	01	0	0	00	280	0	0	1st byte (00) written to ADD=280
533	1	0	C	11	7	1	00	0	0	00	281	0	0	
534	1	0	D	11	E	1	00	0	0	00	281	0	0	
535	1	0	F	11	A	1	00	0	0	00	281	0	0	
536	1	0	E	11	B	1	00	0	0	00	281	0	0	
537	1	0	A	11	9	1	00	0	0	00	281	0	0	
538	1	0	B	11	8	1	00	0	0	00	281	0	0	
539	1	0	9	11	0	1	00	0	0	00	281	0	0	
540	1	0	8	11	1	1	00	0	0	00	281	0	0	
541	1	1	8	01	3	1	00	0	0	00	281	0	0	
542	0	1	0	11	7	1	01	0	0	01	281	0	0	2nd byte (01) written to ADD=281
543	1	0	0	11	7	1	00	0	0	01	282	0	0	
544	1	0	1	11	E	1	00	0	0	02	282	0	0	
545	1	0	3	11	A	1	00	0	0	04	282	0	0	
546	1	0	2	11	B	1	00	0	0	08	282	0	0	
547	1	0	6	11	9	1	00	0	0	10	282	0	0	
548	1	0	7	11	8	1	00	0	0	20	282	0	0	
549	1	0	5	11	0	1	00	0	0	40	282	0	0	
550	1	1	4	11	1	1	00	0	0	80	282	0	0	
551	1	0	0	11	3	1	00	0	0	01	282	0	0	
552	0	0	1	11	7	1	01	0	0	02	282	0	0	3rd byte (02) written to ADD=282
553	1	1	1	11	7	1	00	0	0	02	283	0	0	
554	1	0	0	11	E	1	00	0	0	05	283	0	0	
555	1	0	1	11	A	1	00	0	0	0A	283	0	0	
556	1	0	3	11	B	1	00	0	0	14	283	0	0	
557	1	0	2	11	9	1	00	0	0	28	283	0	0	
558	1	0	6	11	8	1	00	0	0	50	283	0	0	
559	1	0	7	11	0	1	00	0	0	A0	283	0	0	
560	1	1	5	11	1	1	00	0	0	40	283	0	0	
561	1	1	0	11	3	1	00	0	0	81	283	0	0	
562	0	1	0	11	7	1	01	0	0	03	283	0	0	4th byte (03) written to ADD=283
563	1	0	0	11	7	1	00	0	0	03	284	0	0	
564	1	0	1	11	E	1	00	0	0	06	284	0	0	
565	1	0	3	11	A	1	00	0	0	0C	284	0	0	
566	1	0	2	11	B	1	00	0	0	18	284	0	0	
567	1	0	6	11	9	1	00	0	0	30	284	0	0	
568	1	0	7	11	8	1	00	0	0	60	284	0	0	
569	1	1	5	11	0	1	00	0	0	C0	284	0	0	
570	1	0	0	11	1	1	00	0	0	81	284	0	0	
571	1	0	1	11	3	1	00	0	0	02	284	0	0	
572	0	0	3	11	7	1	01	0	0	04	284	0	0	5th byte (04) written to ADD=284
573	1	1	3	11	7	1	00	0	0	04	285	0	0	
574	1	0	0	11	E	1	00	0	0	09	285	0	0	

SEQ	RXCK	DATA	RXSYNDET	SRIXSTYNC	BITCOUNTINV	RXACTSTUS	STATACT	LACTRXER	RXD A<0-7>	ADD	MTE	EN	COMMENTS
575	1	0	1	11	A	1	00	0	0	12	285	0	0	
576	1	0	3	11	B	1	00	0	0	24	285	0	0	
577	1	0	2	11	9	1	00	0	0	48	285	0	0	
578	1	0	6	11	8	1	00	0	0	90	285	0	0	
579	1	1	7	11	0	1	00	0	0	20	285	0	0	
580	1	0	0	11	1	1	00	0	0	41	285	0	0	
581	1	1	1	11	3	1	00	0	0	82	285	0	0	
582	0	1	0	11	7	1	01	0	0	05	285	0	0	6th byte (05) written to ADD=285
583	1	1	0	11	7	1	00	0	0	05	286	0	0	
584	1	0	0	11	E	1	00	0	0	0B	286	0	0	
585	1	0	1	11	A	1	00	0	0	16	286	0	0	
586	1	0	3	11	B	1	00	0	0	2C	286	0	0	
587	1	0	2	11	9	1	00	0	0	58	286	0	0	
588	1	0	6	11	8	1	00	0	0	B0	286	0	0	
589	1	1	7	11	0	1	00	0	0	60	286	0	0	
590	1	1	0	11	1	1	00	0	0	C1	286	0	0	
591	1	0	0	11	3	1	00	0	0	83	286	0	0	
592	0	0	1	11	7	1	01	0	0	06	286	0	0	7th byte (06) written to ADD=286
593	1	1	1	11	7	1	00	0	0	06	287	0	0	
594	1	1	0	11	E	1	00	0	0	0D	287	0	0	
595	1	0	0	11	A	1	00	0	0	1B	287	0	0	
596	1	1	1	11	B	1	00	0	0	36	287	0	0	
597	1	1	0	11	9	1	00	0	0	6D	287	0	0	
598	1	0	0	11	8	1	00	0	0	DB	287	0	0	
599	1	0	1	11	0	1	00	0	0	B6	287	0	0	
600	1	0	3	11	1	1	00	0	0	6C	287	0	0	
601	1	1	2	11	3	1	00	0	0	D8	287	0	0	
602	1	0	0	11	F	0	00	0	0	B1	300	0	0	**8th byte (B1) occurs-SEE MTE3.**
603	1	1	1	11	F	0	00	0	0	62	300	0	0	RXACTIVE RESET-IDLE CHARACTERS IN
604	1	0	0	11	F	0	00	0	0	C5	300	0	1	
605	1	1	1	11	F	0	00	0	0	8A	300	0	1	
606	1	0	0	11	F	0	00	0	0	15	300	0	1	
607	1	1	1	11	F	0	00	0	0	2A	300	0	1	
608	1	0	0	11	F	0	00	0	0	55	300	0	1	
609	1	1	1	11	F	0	00	0	0	AA	300	0	1	
610	1	0	0	11	F	0	00	0	0	55	300	0	1	
611	1	1	1	11	F	0	00	0	0	AA	300	0	1	
612	1	0	0	11	F	0	00	0	0	55	300	0	1	
613	1	1	1	11	F	0	00	0	0	AA	300	0	1	

This file is collected to show the Monitor Threshold Logic operation (MTE). The input for all of these files is the standard 8 byte message ending in a "b1". The data is "0,1,2,3,4,5,6,b1". For this file, Receive Sync detection was ENABLED, the MTE threshold was "6" and the transmit logic was "enabled".

THIS IS A RECEIVE MEMORY STORAGE CYCLE

LOGIC ANALYZER IS SET TO SYNC ON STAT=01

SEQ		RXCK	DATA	RXSYNDET	SRIXXSTYNC	BITCOUNT	RXACTIVE	STATUS	LACT	RXPER	RXDA<0-7>	ADD	MTE	EN	COMMENTS
494		0	0	B	11	3	0	00	1	0	00	300	0	0	LACT = 1, (MTE ALREADY EXCEEDED)
495		0	0	B	11	3	0	00	1	0	00	300	0	0	(ALL DATA WILL BE STORED)
496		0	0	B	11	3	0	00	1	0	00	300	0	0	
497		0	0	B	11	3	0	00	1	0	00	300	0	0	
498		0	0	B	11	3	0	00	1	0	00	300	0	0	
499		0	0	B	11	3	0	00	1	0	00	300	0	0	
500		0	0	B	11	3	0	00	1	0	00	300	0	0	
501		0	0	B	11	3	0	00	1	0	00	300	0	0	
502		0	0	B	11	3	0	00	1	0	00	300	0	0	
503		0	0	B	11	3	0	00	1	0	00	300	0	0	
504		0	0	B	11	3	0	00	1	0	00	300	0	0	
505		0	0	B	11	3	0	00	1	0	00	300	0	0	
506		0	0	B	11	3	0	00	1	0	00	300	0	0	
507		1	0	B	11	3	0	00	1	0	00	300	0	0	RXCK OCCURS
508		0	0	9	11	7	0	00	1	0	00	300	0	0	BITCOUNT=7-GEN LOADRX
509		0	0	9	11	7	0	00	1	1	00	300	0	0	
510	T	0	0	9	11	7	0	01	1	1	00	300	0	0	STATUS->01 (WRITES DATA TO ADD=300)
511		0	0	9	11	7	0	02	1	1	00	300	0	0	STATUS->02
512		0	0	9	11	7	0	00	1	1	00	300	0	0	EN MEMORY ADD COUNTER,STATUS->0
513		0	0	9	11	7	0	00	1	1	00	301	0	0	ADD->301
514		0	0	9	11	7	0	00	1	1	00	301	0	0	
515		0	0	9	11	7	0	00	1	1	00	301	0	0	
516		0	0	9	11	7	0	00	1	1	00	301	0	0	
517		0	0	9	11	7	0	00	1	1	00	301	0	0	
518		0	0	9	11	7	0	00	1	1	00	301	0	0	

THIS IS A RECEIVE MEMORY STATUS STORAGE CYCLE

LOGIC ANALYZER IS SET TO SYNC ON STATUS = 03

SEQ		RXCK	DATA	RXSYNDET	SRIXSTCNT	BITCOUNTINV	RXACT	STATUS	LACT	RXPER	RXDATA<07>	ADDMTE	ME	COMMENTS
502		0	1	2	11	3	1	00	0	0	D8	387	0	0 SYSTEM IS ACTIVE, NO PARITY ERROR
503		0	1	2	11	3	1	00	0	0	D8	387	0	0
504		0	1	2	11	3	1	00	0	0	D8	387	0	0
505		0	1	2	11	3	1	00	0	0	D8	387	0	0
506		0	1	2	11	3	1	00	0	0	D8	387	0	0
507		1	1	2	11	3	1	00	0	0	D8	387	0	0 RXCK OCCURS
508		0	1	0	11	7	1	00	0	0	B1	387	0	0 BITCOUNT=7, LOADRX IS GENERATED
509		0	1	0	11	7	1	00	0	0	B1	387	0	0
510	T	0	1	0	11	7	1	03	0	0	B1	387	0	0 STATUS=3, B1 WRITTEN TO ADD=387
511		0	1	0	11	7	1	04	0	0	B1	387	0	0 STATUS=4, ADD=387 -> LATCH
512		0	1	0	11	7	1	05	0	0	B1	387	0	0 LO_RMAC GEN,Stat->1
513		0	1	0	11	7	1	06	0	0	82	007	0	0 Sel_S0->0
514		0	1	0	11	7	1	07	0	0	00	01C	0	0
515		0	1	0	11	7	1	08	0	0	83	01C	0	0 (83) written to ADD=01C
516		0	1	0	11	7	1	09	0	0	83	01C	0	0 (NO PARITY ERROR,LAST ADD=387)
517		0	1	0	11	7	1	0A	0	0	00	01D	0	0
518		0	1	0	11	7	1	0B	0	0	87	01D	0	0 (87) written to ADD=01D
519		0	1	0	11	7	1	10	0	0	B1	380	0	0 Memory Add Count cleared
520		0	1	0	11	7	1	11	0	0	B1	380	0	0
521		0	1	0	11	F	1	13	0	0	B1	380	0	0 STOP generated
522		0	1	0	11	F	0	00	0	0	B1	080	0	0 ADD->080
523		0	1	0	11	F	0	00	0	0	B1	080	0	0 CYCLE IS COMPLETE
524		0	1	0	11	F	0	00	0	0	B1	080	0	0
525		0	1	0	11	F	0	00	0	0	B1	080	0	0
526		0	1	0	11	F	0	00	0	0	B1	080	0	0

THIS IS A RECEIVE RESET CYCLE (THIS OCCURS WHEN SYNC IS DETECTED)

LOGIC ANALYZER IS SET TO SYNC ON STATUS=0E

SEQ		RXCK	DATA	RXSYNDET	SRIXSTY NC	BITCOUNT	RXACTIVE	STATUS	LACT	RXPER	RXDATA<0-7>	ADDR	MTE	EN	COMMENTS
497		0	1	8	00	F	0	00	0	0	00	081	0	0	
498		0	1	8	00	F	0	00	0	0	00	081	0	0	
499		0	1	8	00	F	0	00	0	0	00	081	0	0	
500		0	1	8	00	F	0	00	0	0	00	081	0	0	
501		0	1	8	00	F	0	00	0	0	00	081	0	0	
502		0	1	8	00	F	0	00	0	0	00	081	0	0	
503		0	1	8	00	F	0	00	0	0	00	081	0	0	
504		0	1	8	00	F	0	00	0	0	00	081	0	0	
505		0	1	8	00	F	0	00	0	0	00	081	0	0	
506		0	1	8	00	F	0	00	0	0	00	081	0	0	
507		0	1	8	00	F	0	00	0	0	00	081	0	0	
508		1	1	8	00	F	0	00	0	0	00	081	0	0	RXCK OCCURS,SIXT=0,RXSYNC=0
509		0	1	0	11	F	1	00	0	0	01	081	0	1	RXACTIVE -> 1,
510	T	0	1	0	11	F	1	0E	0	0	01	081	0	0	STATUS=E,Generates Clr_Rmac
511		0	1	0	11	F	1	00	0	0	01	080	0	0	ADD -> 80
512		0	1	0	11	F	1	00	0	0	01	080	0	0	
513		0	1	0	11	F	1	00	0	0	01	080	0	0	
514		0	1	0	11	F	1	00	0	0	01	080	0	0	

APPENDIX C
PAL SOURCE CODE

FILENAME:ATC2.S

10/4/90 Updated for report
7/30/90 Updated after checkout
6/28/90 Updated before checkout
6/25/90 Original set

Pal Filename	Integrated Ckt #	COMMENTS
DLCONT.ABL	U11	Data Latch Control Logic
ADEC.ABL	U13	VME Address Decode Logic
AINCONT.ABL	U17	Interrupt Control Logic
IPRI2.ABL	U18B	Interrupt Priority Logic
CONOUT.ABL	U19	Control Latch Output Enable
BDEC.ABL	U24	Address Block Decode Logic
CDEC.ABL	U25	Control Address Decode Logic
CLAT.ABL	U26	Control Latch load logic
SLAT.ABL	U27	Hardware Status Latch
INBYT.ABL	U28	Input Byte Control Logic
ACKTIM.ABL	U30	VME Acknowledge Timing Logic
ACKLOG2.ABL	U31B	VME Acknowledge Logic
CLCONT2.ABL	U36B	System Clock Control Logic
INSREG.ABL	U37	Receive Data Input Shift Reg
RXPAR.ABL	U38	Receive Data Parity Check
SYNDET.ABL	U42	Rec. Data Sync Detection
BITCOUNT.ABL	U43	Rec. Data Bit Counter
EOMDET.ABL	U44	Rec. Data End of Message Detection
ZEROMON.ABL	U45	Rec. Data Consecutive Zero Monitor
STATCONA.ABL	U46A,U47	Rec. Data Status Control
RADCON.ABL	U49,U50	Rec. Data Address Control Logic
TXBCT.ABL	U53,U54	Transmit Data Byte Counter
TXCONT.ABL	U55,U56	Transmit Data Control Logic
TXOUTSR.ABL	U57	Transmit Data Output Shift Reg
TXPAR.ABL	U58	Transmit Data Parity Generator
CLKMON.ABL	U60	External Clock Monitor Logic
TXMEMAD.ABL	U61	Trans. Memory Address Counter

```
module DLCONT

This module will control the data bus latches for the
interface between the hardware and the 68020 and
controls the output of the address bus and other
miscellaneous functions. It also provides the multiplex
function required by the VME when swapping the data bit
position for even and odd bytes (bytes 0&2 are D8-D15 and
bytes 1&3 are D0-D7)

title DATA LATCH CONTROL ATC INTERFACE U11

Leo J. Wapelhorst---FAA Tech Center-----6-29-90

   U11 device P22V10;

CLK,HEVN,LEGAL,MAD1                          pin 1,2,4,5;
!WR,LAD,DS,HWACK1,BUSY1                      pin 6,7,8,9,11;
DDIR,DOENE,WRMEM,ACT1                        pin 18,20,15,16;
DOENO                                        pin 17;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;

equations

!DOENE =   HEVN &(!MAD1 & LEGAL & WRMEM  & !DS # !WRMEM) ;
!DOENO =   !HEVN &(!MAD1 & LEGAL & WRMEM  & !DS # !WRMEM) ;
!WRMEM = !ACT1 & WR & !MAD1;
DDIR = WR;
!ACT1 := LEGAL & !MAD1 & !DS #
     !ACT1 & LAD & !DS;
```

```
test_vectors

([CLK,LEGAL,MAD1,DS,LAD,!WR,HWACK1,BUSY1,HEVN] ->
[ACT1,DOENE,DOENO,WRMEM,DDIR])

 WRITE CYCLE-ODD BYTES
[C,0,1,1,0,1,0,0,0] ->  [1,1,1,1,0];
[C,1,0,0,1,1,0,0,0] ->  [0,1,0,1,0]; ACTIVE
[C,1,0,0,1,0,0,0,0] ->  [0,1,0,0,1]; GENERATE ABCK
[C,1,0,0,1,0,1,0,0] ->  [0,1,0,0,1]; REC HWACK1
[C,1,0,0,1,0,1,0,0] ->  [0,1,0,0,1]; WAITING FOR-
[C,1,0,0,1,0,1,0,0] ->  [0,1,0,0,1]; DS & LAD
[C,0,1,1,0,1,0,0,0] ->  [1,1,1,1,0]; CYCLE COMPLETE
"READ CYCLE-ODD BYTES
[C,1,0,1,0,1,0,0,0] ->  [1,1,1,1,0]; WAIT FOR DS/LAD
[C,1,0,0,1,1,0,0,0] ->  [0,1,0,1,0]; ACTIVE
[C,1,0,0,1,1,0,0,0] ->  [0,1,0,1,0]; WAIT FOR HWACK
[C,1,0,0,1,1,1,0,0] ->  [0,1,0,1,0]; REC= >DOEN/BACK
[C,1,0,0,1,1,1,0,0] ->  [0,1,0,1,0]; WAIT FOR DS/LAD
[C,1,0,1,0,1,0,0,0] ->  [1,1,1,1,0]; WAIT HWACK=0
[C,1,0,1,0,1,0,0,0] ->  [1,1,1,1,0]; CYCLE COMPLETE
 WRITE CYCLE-EVEN BYTES
[C,0,1,1,0,1,0,0,1] ->  [1,1,1,1,0];
[C,1,0,0,1,1,0,0,1] ->  [0,0,1,1,0]; ACTIVE
[C,1,0,0,1,0,0,0,1] ->  [0,0,1,0,1]; GENERATE ABCK
[C,1,0,0,1,0,1,0,1] ->  [0,0,1,0,1]; REC HWACK1
[C,1,0,0,1,0,1,0,1] ->  [0,0,1,0,1]; WAITING FOR-
[C,1,0,0,1,0,1,0,1] ->  [0,0,1,0,1]; DS & LAD
[C,0,1,1,0,1,0,0,1] ->  [1,1,1,1,0]; CYCLE COMPLETE
READ CYCLE-EVEN BYTES
[C,1,0,1,0,1,0,0,1] ->  [1,1,1,1,0]; WAIT FOR DS/LAD
[C,1,0,0,1,1,0,0,1] ->  [0,0,1,1,0]; ACTIVE
[C,1,0,0,1,1,0,0,1] ->  [0,0,1,1,0]; WAIT FOR HWACK
[C,1,0,0,1,1,1,0,1] ->  [0,0,1,1,0]; REC= >DOEN/BACK
[C,1,0,0,1,1,1,0,1] ->  [0,0,1,1,0]; WAIT FOR DS/LAD
[C,1,0,1,0,1,0,0,1] ->  [1,1,1,1,0]; WAIT HWACK=0
[C,1,0,1,0,1,0,0,1] ->  [1,1,1,1,0]; CYCLE COMPLETE
end DLCONT
```

```
module ADEC

title ADDRESS DECODER ATC INTERFACE  U13
Leo J. Wapelhorst---FAA Tech Center-----6-14-90
   U13 device P22V10;

LAD,A17,A18,A19,A20,A21                    pin 1,2,3,4,5,6;
A22,A23,A24,A25,A26,A27                    pin 7,8,9,10,11,17;
A16,A28,A29,A30,A31                        pin 18,16,15,14,13;
MAD1                                       pin 23;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
Addin = [A31..A16];

equations

!MAD1 := (Addin = = ^hEF70); "EF7XXXXX

test_vectors ([LAD,Addin] -> [MAD1])

[0,^h0000]      -> [1];
[C,^h0000]      -> [1];
[C,^hFF00]      -> [1];
[C,^hEF70]      -> [0];
[C,^hEF80]      -> [1];
[C,^hEF81]      -> [1];
[C,^hE381]      -> [1];
[C,^h3481]      -> [1];
[C,^hEF81]      -> [1];


end ADEC
```

```
module AINCONT

This PAL controls the interrupt handshaking with the
68020. It receives the requests via INT0-2 from the
priority PAL. It accepts higher level interrupts until
the end of state 1. It then freezes at that vector address
until state 7 when it is reset. It furnishes the priority
PAL with IAK which resets the interrupt request.
The drawing of the state diagram is filename VINTER.FLO.

title INTERRUPT CONTROL VMEINT U17
Leo J. Wapelhorst---FAA Tech Center-----6-27-90
   U17 device P22V10;

CLK,ICY,A1,A2,A3,IAC                                    pin 1,2,3,4,5,6;
DS,LAD,INT0,INT1,INT2,!RESET                            pin 7,8,10,11,9,13;
INTIN                                                   pin 15;
IRQ6,IACKOUT,IAK                                        pin 23,22,14;
VCEN,P,INTACT                                           pin 21,20,19;
VS0,VS1,VS2                                             pin 18,17,16;

reset,preset            node 25,26;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
INT = [INT2,INT1,INT0];
STA = [VCEN,P,IRQ6];
VADD =[VS2,VS1,VS0];

INTACT IStype neg;

output states

S0=^b111;       S1=^b110;       S2=^b010;       S3=^b011;
S4=^b001;       S5=^b000;       S6=^b100;       S7=^b101;

state_diagram [VCEN,P,IRQ6]

        State S0: case !INTACT:S1;
                       endcase;
        State S1: case  [!IAC & !A1&A2&A3 & !ICY]: S2;
                               ![!IAC & !A1&A2&A3 & !ICY]: S1;
                       endcase;
        State S2: GOTO S3;
        State S3: GOTO S4;
        State S4: case  !DS & LAD :S4;
                                DS & LAD  :S4;
                                DS & !LAD :S7;
                       endcase;
        State S5: GOTO S0;
        State S6: GOTO S0;
        State S7: GOTO S0;
```

```
equations

reset = RESET;

IACKOUT = IAC # ICY # (!IRQ6 & !A1 & A2 & A3 & !IAC) #
      !VCEN # (STA = = 5);

IAK = (STA = = ^h3);STATE 3

!INTACT := INTIN  # !INTACT & !(STA = = 5);

!VS2 := (STA = = 6) & !INT2 # !VS2 & !(STA = = 5);
!VS1 := (STA = = 6) & !INT1 # !VS1 & !(STA = = 5);
!VS0 := (STA = = 6) & !INT0 # !VS0 & !(STA = = 5);
```

```
test_vectors
([!RESET,CLK,IAC,ICY,A1,A2,A3,INT2,INT1,INT0,DS,LAD,INTIN] ->
[IACKOUT,IAK,VCEN,P,IRQ6,INTACT,VS2,VS1,VS0])

[0,C,1,1,X,X,X,1,1,1,1,0,0]-> [1,0,1,1,1,1,1,1,1];
[0,C,1,1,X,X,X,1,1,1,1,0,0]-> [1,0,1,1,1,1,1,1,1];
[1,C,1,1,X,X,X,1,1,1,1,0,0]-> [1,0,1,1,1,1,1,1,1];
[1,C,1,1,X,X,X,1,1,1,1,0,0]-> [1,0,1,1,1,1,1,1,1];
[1,C,1,1,X,X,X,1,1,1,1,0,0]-> [1,0,1,1,1,1,1,1,1];
[1,C,1,1,X,X,X,1,1,0,1,0,1]-> [1,0,1,1,1,0,1,1,1];ILEVEL 1
[1,C,1,1,X,X,X,1,1,0,1,0,1]-> [1,0,1,1,0,0,1,1,1];IRQ GEN
[1,C,1,1,X,X,X,1,1,0,1,0,1]-> [1,0,1,1,0,0,1,1,0];state 1
[1,C,1,0,X,X,X,1,1,0,1,0,1]-> [1,0,1,1,0,0,1,1,0];state 1
[1,C,0,0,0,1,1,1,1,0,0,1,1]-> [1,0,0,1,0,0,1,1,0];ack sta=2
[1,C,0,0,0,1,1,1,1,0,0,1,1]-> [1,1,0,1,1,0,1,1,0];sta=3 IAK
[1,C,0,0,0,1,1,1,1,0,0,1,0]-> [1,0,0,0,1,0,1,1,0];state 4
[1,C,0,0,0,1,1,1,1,1,0,1,0]-> [1,0,0,0,1,0,1,1,0];state 4
[1,C,0,0,0,1,1,1,1,1,1,1,0]-> [1,0,0,0,1,0,1,1,0];state 4
[1,C,0,0,0,1,1,1,1,1,1,0,0]-> [1,0,1,0,1,0,1,1,0];state 7
[1,C,1,1,X,X,X,1,1,1,1,0,0]-> [1,0,1,1,1,1,1,1,1];sta=0-v16
higher level interrupt (6)received during state1
[1,C,1,1,X,X,X,0,1,1,1,0,1]-> [1,0,1,1,1,0,1,1,1];ILEVEL 4
[1,C,1,1,X,X,X,0,1,1,1,0,1]-> [1,0,1,1,0,0,1,1,1];IRQ GEN
[1,C,1,1,X,X,X,0,1,1,1,0,1]-> [1,0,1,1,0,0,0,1,1];vadd=4
[1,C,1,1,X,X,X,0,1,1,1,0,1]-> [1,0,1,1,0,0,0,1,1];
[1,C,1,1,X,X,X,0,0,1,1,0,1]-> [1,0,1,1,0,0,0,0,1];vadd=6
[1,C,1,0,X,X,X,0,0,1,1,0,1]-> [1,0,1,1,0,0,0,0,1];state 1
[1,C,0,0,0,1,1,0,0,1,0,1,1]-> [1,0,0,1,0,0,0,0,1];ack sta=2
[1,C,0,0,0,1,1,1,1,1,0,1,1]-> [1,1,0,1,1,0,0,0,1];sta=3 IAK
[1,C,0,0,0,1,1,1,1,1,0,1,1]-> [1,0,0,0,1,0,0,0,1];state 4
[1,C,0,0,0,1,1,1,1,1,0,1,1]-> [1,0,0,0,1,0,0,0,1];state 4
[1,C,0,0,0,1,1,1,1,1,1,1,1]-> [1,0,0,0,1,0,0,0,1];state 4
[1,C,0,0,0,1,1,1,1,1,1,0,1]-> [1,0,1,0,1,0,0,0,1];state 7
[1,C,1,1,X,X,X,1,1,1,1,0,1]-> [1,0,1,1,1,0,1,1,1];state 0
[1,C,1,1,X,X,X,0,1,1,1,0,1]-> [1,0,1,1,0,0,1,1,1];I4-V30
[1,C,1,1,X,X,X,0,1,1,1,0,1]-> [1,0,1,1,0,0,0,1,1];IRQ GEN
[1,C,1,1,X,X,X,0,1,1,1,0,1]-> [1,0,1,1,0,0,0,1,1];state 1
[1,C,1,0,X,X,X,0,1,1,1,0,1]-> [1,0,1,1,0,0,0,1,1];state 1
[1,C,0,0,0,1,1,0,1,1,0,1,1]-> [1,0,0,1,0,0,0,1,1];ack sta=2
[1,C,0,0,0,1,1,1,1,1,0,1,1]-> [1,1,0,1,1,0,0,1,1];sta=3 IAK
[0,C,1,1,X,X,X,1,1,1,1,0,0]-> [1,0,1,1,1,1,1,1,1];reset

end AINCONT
```

```
module IPRI2

title INTERRUPT PRIORITY LOGIC -ATCINT- U18B
Leo J. Wapelhorst---FAA Tech Center-----7-7-90
   U18B device P22V10;

This Pal passes the interrupt requests on to the interrupt
control pal (aincont.abl). Transmit interrupts
(TINT) have priority over receive (RINT). The interrupt
requests are active high and remain set until the interrupt
is serviced. The vector request is an ACTIVE
LOW three bit field (IRQA,IRQB,IRQC). The outputs are also
active low (INT2,INT1,INT0). All interrupts are
cleared when a RESET is received. A separate vector is
generated for each block of addresses of receive data.

The following table shows the relationship between the
addresses and the vectors.

Mem group     Rx Mem Add    AD7-9  Int0-2   Pr Dat
  1      EF701080            02      02              D0(vector)
  2      EF701100            03      03              D1(vector)
  3      EF701180            04      04              D2(vector)
  4      EF701200            05      05              D3(vector)
  5      EF701280            06      06              D4(vector)
  6      EF701300            07      07              D5(vector)
  7      EF701380            01      01              D6(vector)
TRANSMIT VECTOR                  00          D7


CLK, IREQA,IREQB,IREQC          pin 1,2,3,4;
VS0,VS1,VS2                  pin 5,6,7;
IAK,SRESET,TINT,RINT            pin 9,13,8,10;
DISINT                      pin 11;
INT2,INT1,INT0                pin 23,22,21;
I7,I6,I5,I4,I3,I2,I1          pin 20,19,18,17,16,15,14;

reset,preset                node 25,26;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
IIN     = [IREQC,IREQB,IREQA];
IOUT     = [INT2,INT1,INT0];
VEC     = [VS2,VS1,VS0];

equations

reset = !SRESET;

INT2 = IREQC & !TINT;
INT1 = IREQB & !TINT;
INT0 = IREQA & !TINT;


end IPRI2
```

```
module CONOUT

title CONTROL LATCH OUTPUT ENABLE - ATC INTERFACE  U19
Leo J. Wapelhorst---FAA Tech Center-----6-18-90
   U19 device P22V10;

CLK,DEC0,DEC01,DEC02,DEC03                    pin 1,2,3,4,5;
DEC04,DEC10,ETIME                             pin 6,7,13;
EN0OUT,EN01OUT,EN02OUT                        pin 23,22,21;
EN03OUT,EN04OUT,EN10OUT                       pin 20,19,18;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;

equations

!EN0OUT  = !DEC0 & ETIME;
!EN01OUT = !DEC01 & ETIME;
!EN02OUT = !DEC02 & ETIME;
!EN03OUT = !DEC03 & ETIME;
!EN04OUT = !DEC04 & ETIME;
!EN10OUT = !DEC10 & ETIME;

test_vectors

([DEC0,DEC01,DEC02,DEC03,DEC04,ETIME,DEC10] ->
[EN0OUT,EN01OUT,EN02OUT,EN03OUT,EN04OUT,EN10OUT])

[1,1,1,1,1,0,1] -> [1,1,1,1,1,1];
[1,1,1,1,1,1,1] -> [1,1,1,1,1,1];
[0,1,1,1,1,1,1] -> [0,1,1,1,1,1];
[1,0,1,1,1,1,1] -> [1,0,1,1,1,1];
[1,1,0,1,1,1,1] -> [1,1,0,1,1,1];
[1,1,1,0,1,1,1] -> [1,1,1,0,1,1];
[1,1,1,1,0,0,1] -> [1,1,1,1,1,1];
[1,1,1,1,0,1,1] -> [1,1,1,1,0,1];
[1,1,1,1,1,1,0] -> [1,1,1,1,1,0];

end CONOUT
```

```
module BDEC

title ADDRESS BLOCK DECODER ATC INTERFACE  U24
Leo J. Wapelhorst---FAA Tech Center-----6-18-90
   U24 device P22V10;

MAD1,AD8,AD9,AD10,AD11                          pin 1,2,3,4,5;
AD12,AD13,AD14,AD15,!WR                         pin 6,7,8,9,10;
CONBLK,TXBLK,TXLPOE                             pin 23,22,21;
RXBLK,RXLPOE                                    pin 20,19;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
Addin = [AD15..AD8];

This macro will return a L if the address (a) is between
the two limits (b,c), otherwise it returns a H
between macro (a,b,c)
        {@if((?a>=?b)&(?a<=?c)){L}@if!((?a>=?b)&(?a<=?c)){H}};

equations

!CONBLK= !MAD1 & (Addin==^h00); EF700000-EF700100
!RXBLK = !MAD1&(Addin>=^h10)&(Addin<^h14);EF701000-EF7013ff
!TXBLK = !MAD1&(Addin>=^h20)&(Addin<^h24);EF702000-EF7023ff
!RXLPOE  = !RXBLK & !WR;
!TXLPOE  = !TXBLK & !WR;

test_vectors

([MAD1,Addin] -> [CONBLK,RXBLK,TXBLK])
@radix 16;
@const addrs = 0;
@repeat 100
        {[0,addrs] -> [between (addrs,^h0,^h0),   CONBLK
            between (addrs,^h10,^h13), RXBLK
            between (addrs,^h20,^h23)];TXBLK
@const addrs = addrs +1;}

@const addrs = 0;
@repeat 100
        {[1,addrs] -> [1,1,1];  NOT THE EF700XXXX BLOCK
@const addrs = addrs +1;}

@radix ^hA;

end BDEC
```

```
module CDEC

title CONTROL ADDRESS DECODER ATC INTERFACE  U25
Leo J. Wapelhorst---FAA Tech Center-----6-18-90
   U25 device P22V10;

CONBLK,AD0,AD1,AD2,AD3                          pin 1,2,3,4,5;
AD4,AD5,AD6,AD7                                 pin 6,7,8,9;
DEC0,DEC01,DEC02,DEC03                          pin 23,22,21,20;
DEC04,DEC10,DEC05                               pin 19,18,17;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
Addin = [AD7..AD0];

This macro will return a L if the address (a) is between
the two limits (b,c), otherwise it returns a H
between macro (a,b,c)
        {@if((?a> =?b)&(?a< =?c)){L}@if!((?a> =?b)&(?a< =?c)){H}};

equations

!DEC0  = !CONBLK & (Addin= =^h00); EF700000
!DEC01 = !CONBLK & (Addin= =^h01); EF700001
!DEC02 = !CONBLK & (Addin= =^h02); EF700002
!DEC03 = !CONBLK & (Addin= =^h03); EF700003
!DEC04 = !CONBLK & (Addin= =^h04); EF700004
!DEC05 = !CONBLK & (Addin= =^h05); EF700005
!DEC10 = !CONBLK & (Addin= =^h10); EF700010

test_vectors

([CONBLK,Addin] -> [DEC0,DEC01,DEC02,DEC03,DEC04,DEC05,DEC10])
@radix 16;
@const addrs = 0;
@repeat 100
        {[0,addrs] -> [between (addrs,^h0,^h0), DEC0
            between (addrs,^h1,^h1), DEC01
            between (addrs,^h2,^h2), DEC02
            between (addrs,^h3,^h3), DEC03
            between (addrs,^h4,^h4), DEC04
            between (addrs,^h5,^h5), DEC05
            between (addrs,^h10,^h10)];DEC10
@const addrs = addrs +1;}

@const addrs = 0;
@repeat 100
        {[1,addrs] -> [1,1,1,1,1,1,1]; NOT THE EF700XXXX BLOCK
@const addrs = addrs +1;}

@radix ^hA;

end CDEC
```

```
module CLAT

title CONTROL ADDRESS LATCH CONTROL - ATC INTERFACE  U26
Leo J. Wapelhorst---FAA Tech Center-----6-18-90
   U26 device P22V10;

CLK,DEC0,DEC01,DEC02,DEC03                    pin 1,2,3,4,5;
DEC04,DEC10,LTIME                             pin 6,7,13;
LAT0,LAT01,LAT02,LAT03                        pin 23,22,21,20;
LAT04                                         pin 19;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;

equations

LAT0  = !DEC0 & LTIME;
LAT01 = !DEC01 & LTIME;
LAT02 = !DEC02 & LTIME;
LAT03 = !DEC03 & LTIME;
LAT04 = !DEC04 & LTIME;

test_vectors

([DEC0,DEC01,DEC02,DEC03,DEC04,LTIME] -> [LAT0,LAT01,LAT02,LAT03,LAT04])

[1,1,1,1,1,0] -> [0,0,0,0,0];
[1,1,1,1,1,1] -> [0,0,0,0,0];
[0,1,1,1,1,1] -> [1,0,0,0,0];
[1,0,1,1,1,1] -> [0,1,0,0,0];
[1,1,0,1,1,1] -> [0,0,1,0,0];
[1,1,1,0,1,1] -> [0,0,0,1,0];
[1,1,1,1,0,0] -> [0,0,0,0,0];
[1,1,1,1,0,1] -> [0,0,0,0,1];

end CLAT
```

```
module SLAT

title STATUS LATCH - ATC INTERFACE  U27
Leo J. Wapelhorst---FAA Tech Center-----6-18-90
   U27 device P22V10;

CLK,EN10OUT,NOTXCK,NORXCK                    pin 1,2,3,4;
TXACTIVE,RXACTIVE                            pin 5,6;
DB0,DB1,DB2,DB3                              pin 22,21,20,19;
DB4,DB5,DB6,DB7                              pin 18,17,16,15;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
DAT      = [DB7..DB0];
DAT_re   = [DB7.re,DB6.re,DB5.re,DB4.re,DB3.re,DB2.re,DB1.re,DB0.re];
DAT_oe   = [DB7.oe,DB6.oe,DB5.oe,DB4.oe,DB3.oe,DB2.oe,DB1.oe,DB0.oe];
equations

DAT_oe   = !EN10OUT;
DB7    := NOTXCK;
DB6    := NORXCK;
DB5    := TXACTIVE;
DB4    := RXACTIVE;
DB3     = L;
DB2     = L;
DB1     = L;
DB0     = L;
test_vectors


([CLK,EN10OUT,NOTXCK,NORXCK,TXACTIVE,RXACTIVE] -> [DAT])

[C,0,1,1,0,1] -> [^hD0];
[C,0,1,0,0,1] -> [^h90];
[C,0,0,1,0,1] -> [^h50];
[C,1,1,1,0,1] -> [Z];
[C,0,1,1,0,1] -> [^hD0];

end SLAT
```

```
module INBYT

SRESET controls the logic of the ATC interface card. The
hardware is disabled until a write is made to address
EF700005. The hardware then remains enabled until a 'read'
to the same address which again resets all the hardware
until another write occurs.

title INPUT BYTE CONTROL LOGIC -- ATCINT U28
Leo J. Wapelhorst---FAA Tech Center-----6-28-90
   U28 device E0600;

CLK,!DS,!LWORD,!DS0,!DS1                    pin  1,13,23,22,21;
A01,!WRITE,IACKIN                           pin  20,19,14;
!RESET,DEC05                                pin  2,11;
HAD0,HAD1,BEN,SRESET                        pin  3,4,5,6;
HEVN,!WR,IAC                                pin  18,7,8;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
DIN   = [!DS1,!DS0,A01,!LWORD];
AOUT  = [HAD1,HAD0];

equations

!WR   := !WRITE;
IAC   := IACKIN;
!SRESET = RESET # !RESET&!WR&!DEC05 # !RESET&!SRESET&DEC05;
BEN    = (DIN==^h5) # (DIN==^h7) # (DIN==^h9) # (DIN==^hB);

truth_table

([!DS1,!DS0,A01,!LWORD] :> [AOUT,HEVN])
[0,   1, 0,  1  ]:> [ 0  , 1 ];
[0,   1, 1,  1  ]:> [ 2  , 1 ];
[1,   0, 0,  1  ]:> [ 1  , 0 ];
[1,   0, 1,  1  ]:> [ 3  , 0 ];
```

```
test_vectors
([CLK,!DS,DIN,!RESET,DEC05,IACKIN,!WRITE]-> [HEVN,AOUT,!WR,SRESET,IAC,BEN])

[C,C,^h0,0,X,0,0] -> [X,  X,0,0,0,0];
[C,C,^h1,1,1,0,0] -> [X,  X,0,0,0,0];
[C,C,^h2,1,1,0,0] -> [X,  X,0,0,0,0];
[C,C,^h3,1,1,0,0] -> [X,  X,0,0,0,0];
[C,C,^h4,1,1,0,0] -> [X,  X,0,0,0,0];
[C,C,^h5,1,1,0,0] -> [1,^h0,0,0,0,1];
[C,C,^h6,1,1,0,0] -> [X,  X,0,0,0,0];
[C,C,^h7,1,1,1,0] -> [1,^h2,0,0,1,1];
[C,C,^h8,1,1,1,0] -> [X,  X,0,0,1,0];
[C,C,^h9,1,0,0,0] -> [0,^h1,0,1,0,1];DEC05-ENABLE
[C,C,^hA,1,1,0,0] -> [X,  X,0,1,0,0];
[C,C,^hB,1,1,0,0] -> [0,^h3,0,1,0,1];
[C,C,^hC,1,1,0,0] -> [X,  X,0,1,0,0];
[C,C,^hD,1,1,0,0] -> [X,  X,0,1,0,0];
[C,C,^hE,1,1,0,0] -> [X,  X,0,1,0,0];
[C,C,^hF,1,1,0,1] -> [X,  X,1,1,0,0];
[C,C,^h9,1,0,0,1] -> [0,^h1,1,0,0,1];DEC05-RESET
[C,C,^h1,1,1,0,1] -> [X,  X,1,0,0,0];
[C,C,^h2,1,1,0,0] -> [X,  X,0,0,0,0];
[C,C,^h3,1,1,0,0] -> [X,  X,0,0,0,0];
[C,C,^h4,1,1,0,0] -> [X,  X,0,0,0,0];
[C,C,^h9,1,0,0,0] -> [0,^h1,0,1,0,1];DEC05-ENABLE
[C,C,^h0,0,X,0,0] -> [X,  X,0,0,0,0];RESET

end INBYT
```

```
module ACKTIM

title I/O ACKNOWLEDGE TIMING - ATC INTERFACE U30
Leo J. Wapelhorst---FAA Tech Center----6-19-90
   U30 device P22V10;

CLK,MAD1,LAD,DS,!WR                                    pin 1,2,3,4,5;
DEC10,TXBLK,TMBUSY,RXBLK,RMBUSY                        pin 6,7,8,9,10;
ETIME,LTIME,DSTRDLY                                    pin 23,22,18
;
HWACK,TXLPWR,RXLPWR            pin 21,20,19;
reset,preset                   node  25,26;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;

equations

DSTRDLY := !DS;
LTIME  := !DS & !DSTRDLY & WR & !MAD1;
ETIME  = !DS & !WR & !MAD1;
HWACK   = (DSTRDLY & !DS & !MAD1 & (TMBUSY & !TXBLK)) #
       (DSTRDLY & !DS & !MAD1 & (RMBUSY & !RXBLK)) #
       (DSTRDLY & !DS & !MAD1 & !DEC10 & !WR) #"READ 010
       (DSTRDLY & !DS & !MAD1 & DEC10 & TXBLK & RXBLK);
!TXLPWR := !DS & !TXBLK & WR & !DSTRDLY;
!RXLPWR := !DS & !RXBLK & WR & !DSTRDLY;
```

```
test_vectors

([MAD1,CLK,LAD,DS,!WR,TXBLK,RXBLK,TMBUSY,DEC10] ->
[ETIME,TXLPWR,RXLPWR,LTIME,HWACK,DSTRDLY])

[1,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,X];
[1,C,1,0,1,1,1,1,1] -> [0,1,1,0,0,1];READ - NOT MINE
[1,C,1,0,1,1,1,1,1] -> [0,1,1,0,0,1];READ - NOT MINE
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];CY COMPLETE
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,1,0,1,1,1,1,1] -> [1,1,1,0,1,1];READ - NOT MEM
[0,C,1,0,1,1,1,1,1] -> [1,1,1,0,1,1];
[0,C,1,0,1,1,1,1,1] -> [1,1,1,0,1,1];
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];CY COMPLETE
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,1,0,1,0,1,1,1] -> [1,1,1,0,1,1];READ - TXBLK
[0,C,1,0,1,0,1,1,1] -> [1,1,1,0,1,1];
[0,C,1,0,1,0,1,1,1] -> [1,1,1,0,1,1];READ - ACK
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,1,0,0,0,1,1,1] -> [0,0,1,1,1,1];TX MEM WRITE
[0,C,1,0,0,0,1,1,1] -> [0,1,1,0,1,1];VECTOR 20
[0,C,1,0,0,0,1,1,1] -> [0,1,1,0,1,1];
[0,C,1,0,0,0,1,1,1] -> [0,1,1,0,1,1];
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];CYCLE COMPLETE
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];
[0,C,1,0,0,1,1,1,1] -> [0,1,1,1,1,1];WRITE - NOT MEM
[0,C,1,0,0,1,1,1,1] -> [0,1,1,0,1,1];LATCH DATA,DACK
[0,C,1,0,0,1,1,1,1] -> [0,1,1,0,1,1];
[0,C,0,1,1,1,1,1,1] -> [0,1,1,0,0,0];CY COMPLETE

end ACKTIM
```

```
module ACKLOG2

This module will generate the acknowledge signals required
to communicate with the 68020 system. This chip also
receives the interrupt requests from the transmit and
receive control logic and stores the fact that one in in
progress while it waits for the interrupt logic of U17 and
U18 to process the request. It resets the logic when an
IAK is received with the proper interrupt vector (VS0,1,2).


title ACK LOGIC ATC INTERFACE U31B
Leo J. Wapelhorst---FAA Tech Center-----7-7-90
U31B device P22V10;

CLK,ACT1,IAK,RXINT,TXINT,HWACK                    pin 1,2,3,4,5,7;
VS0,VS1,VS2,SRESET                                pin 8,9,10,13;
HWACK1,BUSY1,DATACK,BUSERR                        pin 23,21,18,17;
INTIN,TINT,RINT                                   pin 16,15,14;

reset,preset            node 25,26;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
VSEL = [VS2,VS1,VS0];

equations


HWACK1 := HWACK;
BUSERR := L;
DATACK := IAK  # HWACK & !ACT1;
BUSY1  := ACT1;
TINT   := SRESET &
      (TXINT & !TINT  # TINT & !IAK # TINT & !(VSEL==0));
RINT   := SRESET &
      (RXINT & !RINT  # RINT & !IAK # RINT &  (VSEL==0));
INTIN  := SRESET & ((TINT # RINT) & !IAK);
```

```
test_vectors

([SRESET,CLK,ACT1,VSEL,IAK,HWACK,RXINT,TXINT] ->
[INTIN,HWACK1,BUSY1,DATACK,BUSERR,TINT,RINT])

[0,C,0,^h0,0,0,0,0]     ->    [X,0,0,0,0,0,0];   RESET
[1,C,0,^h0,0,0,0,0]     ->    [0,0,0,0,0,0,0];   ENABLED
[1,C,1,^h6,0,0,0,1]     ->    [0,0,1,0,0,1,0];   TXINT
[1,C,1,^h0,0,0,0,0]     ->    [1,0,1,0,0,1,0];
[0,C,0,^h0,0,0,0,0]     ->    [0,0,0,0,0,0,0];   RESET
[1,C,1,^h5,0,0,0,0]     ->    [0,0,1,0,0,0,0];
[1,C,1,^h0,0,0,0,1]     ->    [0,0,1,0,0,1,0];   TXINT
[1,C,0,^h0,0,0,0,0]     ->    [1,0,0,0,0,1,0];
[0,C,0,^h0,0,0,0,0]     ->    [0,0,0,0,0,0,0];   reset CLRS
[1,C,1,^h3,0,0,1,0]     ->    [0,0,1,0,0,0,1];   INT4
[1,C,1,^h0,0,0,0,0]     ->    [1,0,1,0,0,0,1];
[1,C,0,^h0,1,0,0,0]     ->    [0,0,0,1,0,0,1];   IAK - NO MATCH
[1,C,1,^h0,0,0,0,0]     ->    [1,0,1,0,0,0,1];
[1,C,1,^h2,1,0,0,0]     ->    [0,0,1,1,0,0,0];   IACK RESETS INT
[1,C,1,^h3,0,0,0,0]     ->    [0,0,1,0,0,0,0];
[1,C,1,^h3,0,0,1,0]     ->    [0,0,1,0,0,0,1];
[1,C,1,^h0,0,0,0,0]     ->    [1,0,1,0,0,0,1];
[1,C,1,^h2,0,0,0,0]     ->    [1,0,1,0,0,0,1];   INT 5
[1,C,0,^h2,1,0,0,0]     ->    [0,0,0,1,0,0,0];   IACK CLRS,DTACK GEN
[1,C,0,^h0,0,0,0,0]     ->    [0,0,0,0,0,0,0];
[1,0,0,^h0,0,1,0,0]  ->    [0,0,0,0,0,0,0];    HWACK OCCURS
[1,C,0,^h0,0,1,0,0]     ->    [0,1,0,1,0,0,0];   DTACK GEN
[1,C,0,^h0,0,1,0,0]     ->    [0,1,0,1,0,0,0];
[1,C,0,^h0,0,1,0,0]     ->    [0,1,0,1,0,0,0];
[1,C,0,^h0,0,1,0,0]     ->    [0,1,0,1,0,0,0];
[0,C,0,^h0,0,0,0,0]     ->    [0,0,0,0,0,0,0];   DTACK RESET
SRESET ACTIVE
[0,C,0,^h0,0,0,0,0]     ->    [0,0,0,0,0,0,0];
[0,0,0,^h0,0,1,0,0]->    [0,0,0,0,0,0,0];    HWACK OCCURS
[0,C,0,^h0,0,1,0,0]     ->    [0,1,0,1,0,0,0];   DTACK GEN
[0,C,0,^h0,0,1,0,0]     ->    [0,1,0,1,0,0,0];
[0,C,0,^h0,0,1,0,0]     ->    [0,1,0,1,0,0,0];
[0,C,0,^h0,0,1,0,0]     ->    [0,1,0,1,0,0,0];
[0,C,0,^h0,0,0,0,0]     ->    [0,0,0,0,0,0,0];   DTACK RESET
[1,C,0,^h0,0,0,0,0]     ->    [0,0,0,0,0,0,0];   ENABLED
[1,C,1,^h0,0,0,0,1]     ->    [0,0,1,0,0,1,0];   TXINT
[1,C,1,^h0,0,0,0,0]     ->    [1,0,1,0,0,1,0];
[1,C,1,^h0,1,0,0,0]     ->    [0,0,1,1,0,0,0];   IACK CLEARS
[1,C,1,^h0,0,0,0,0]     ->    [0,0,1,0,0,0,0];
[1,C,1,^h0,0,0,0,0]     ->    [0,0,1,0,0,0,0];

end ACKLOG2
```

```
module CLCONT2

This module will use a counter to eliminate glitches which
occur within 600 ns of each other

title CLOCK CONTROL MODULE- ATC INTERFACE  U36B
Leo J. Wapelhorst---FAA Tech Center-----10-4-90
   U36B device P22V10;

CLK,!EXRXCLK,EXTXCLK,ICLK                    pin 1,2,3,4;
USEXTRX,USEXTTX,RECDATA                      pin 5,6,7;
INVDAT,TXDATA,LOOPBK                         pin 8,9,10;
DISRX,DISTX                                  pin 11,13;
TXCKEN,RXCKEN,RXDVAR                         pin 23,22,21;
TXDOUT                                       pin 14;
RC2,RC1,RC0                                  pin 20,19,18;
TC2,TC1,TC0                                  pin 17,16,15;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
RCT     = [RC2,RC1,RC0];
TCT     = [TC2,TC1,TC0];
equations

RCT:=((USEXTRX&EXRXCLK#ICLK&!USEXTRX)) & (RCT+1)&(RCT!=7) #
   ((USEXTRX&EXRXCLK#ICLK&!USEXTRX)) & (RCT==7);

RXCKEN := (RCT==6) & (USEXTRX&EXRXCLK#ICLK&!USEXTRX);

TCT:=((USEXTTX&EXTXCLK#!ICLK&!USEXTTX)) & (TCT+1)&(TCT!=7) #
   ((USEXTTX&EXTXCLK#!ICLK&!USEXTTX)) & (TCT==7);

TXCKEN := (TCT==6) & (USEXTTX&EXTXCLK#!ICLK&!USEXTTX);

RXDVAR     = (RECDATA $ INVDAT) & !LOOPBK #
        TXDATA & LOOPBK;
TXDOUT     = TXDATA $ INVDAT;
```

test_vectors

([CLK,USEXTRX,EXRXCLK,ICLK,DISRX,INVDAT,RECDATA] -> [RXCKEN,RXDVAR,RCT])

[C,1,0,X,1,1,1] -> [0,0,0];DISABLE RX,USE EXT RX
[C,1,0,X,1,1,1] -> [0,0,0];RXCKEN DISABLED
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,1,X,0,1,1] -> [0,0,1];
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,1,X,0,1,0] -> [0,1,1];
[C,1,0,X,0,1,0] -> [0,1,0];
[C,1,1,X,0,1,0] -> [0,1,1];
[C,1,1,X,0,1,0] -> [0,1,2];
[C,1,1,X,0,1,0] -> [0,1,3];
[C,1,1,X,0,1,0] -> [0,1,4];
[C,1,1,X,0,1,0] -> [0,1,5];
[C,1,1,X,0,1,0] -> [0,1,6];
[C,1,1,X,0,1,0] -> [1,1,7];RXCKEN
[C,1,1,X,0,1,0] -> [0,1,7];[C,1,1,X,0,1,0] -> [0,1,7];"
[C,1,1,X,0,1,0] -> [0,1,7];
[C,1,1,X,0,1,0] -> [0,1,7];
[C,1,0,X,0,1,1] -> [0,0,0];TE INPUT RX CLK
[C,1,1,X,0,1,1] -> [0,0,1];
[C,1,1,X,0,1,1] -> [0,0,2];
[C,1,1,X,0,1,1] -> [0,0,3];
[C,1,1,X,0,1,1] -> [0,0,4];
[C,1,1,X,0,1,1] -> [0,0,5];
[C,1,1,X,0,1,1] -> [0,0,6];
[C,1,0,X,0,1,1] -> [0,0,0];RXCKEN BLOCKED
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,0,X,0,1,1] -> [0,0,0];
[C,1,0,X,0,1,1] -> [0,0,0];
[C,0,X,0,0,1,1] -> [0,0,0];USE INT RX CLK V25
[C,0,X,1,0,0,1] -> [0,1,1];USE INT CLK-DO NOT INVERT DATA
[C,0,X,1,0,0,1] -> [0,1,2];
[C,0,X,1,0,0,1] -> [0,1,3];
[C,0,X,1,0,0,1] -> [0,1,4];
[C,0,X,1,0,0,1] -> [0,1,5];
[C,0,X,1,0,0,1] -> [0,1,6];
[C,0,X,1,0,0,1] -> [1,1,7];RXCKEN
[C,0,X,1,0,0,1] -> [0,1,7];
[C,0,X,1,0,0,1] -> [0,1,7];
[C,0,X,1,0,0,1] -> [0,1,7];
[C,0,X,1,0,0,1] -> [0,1,7];
[C,0,X,1,0,0,1] -> [0,1,7];
[C,0,X,1,0,0,1] -> [0,1,7];
[C,1,1,X,1,1,1] -> [0,0,7];DIS RX
[C,1,1,X,1,1,1] -> [0,0,7];V40
[C,1,1,X,1,1,1] -> [0,0,7];

```
[C,1,0,X,1,1,1] -> [0,0,0];
[C,1,0,X,1,1,0] -> [0,1,0];
[C,1,0,X,1,1,0] -> [0,1,0];
[C,1,1,X,1,1,0] -> [0,1,1];
[C,1,1,X,1,1,0] -> [0,1,2];
[C,1,1,X,1,1,0] -> [0,1,3];
[C,1,0,X,1,1,1] -> [0,0,0];
[C,1,0,0,0,1,1] -> [0,0,0];EN RX
[C,1,1,1,0,0,1] -> [0,1,1];USE EXT CLK-DO NOT INVERT DATA
[C,1,1,1,0,0,1] -> [0,1,2];
[C,1,1,1,0,0,1] -> [0,1,3];
[C,1,1,1,0,0,1] -> [0,1,4];
[C,1,1,1,0,0,1] -> [0,1,5];
[C,1,1,1,0,0,1] -> [0,1,6];
[C,1,1,1,0,0,1] -> [1,1,7];
[C,0,1,1,0,0,1] -> [0,1,7];

end CLCONT2
```

```
module INSREG

title INPUT SHIFT REGISTER ATCINT U37
        Leo J. Wapelhorst---FAA Tech Center-----6-20-90

        U37 device P22V10;

CLK,RXCKEN,RXDVAR                                   pin 1,2,3;
STAT                                                pin 11;
D0,D1,D2,D3,D4,D5,D6,D7                             pin 23,22,21,20,19,18,17,16;
RXPARIN                                             pin 15;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
Dout = [D7,D6,D5,D4,D3,D2,D1,D0];
Dout_oe = [D7.oe,D6.oe,D5.oe,D4.oe,D3.oe,D2.oe,D1.oe,D0.oe];

equations

RXPARIN := RXCKEN & D7 # RXPARIN & !RXCKEN;
D7    := RXCKEN & D6 # D7 & !RXCKEN;
D6    := RXCKEN & D5 # D6 & !RXCKEN;
D5    := RXCKEN & D4 # D5 & !RXCKEN;
D4    := RXCKEN & D3 # D4 & !RXCKEN;
D3    := RXCKEN & D2 # D3 & !RXCKEN;
D2    := RXCKEN & D1 # D2 & !RXCKEN;
D1    := RXCKEN & D0 # D1 & !RXCKEN;
D0    := RXCKEN & RXDVAR # D0 & !RXCKEN;
Dout_oe  = !STAT;

test_vectors

([CLK,RXDVAR,STAT,RXCKEN] -> Dout)
[       0,      0,      0,      1]      -> 0;
[       C,      0,      0,      1]      -> 0;
[       0,      1,      0,      1]      -> 0;
[       C,      1,      0,      1]      -> 1;
[       C,      1,      0,      1]      -> 3;
[       C,      1,      0,      1]      -> 7;
[       C,      1,      0,      1]      -> ^hf;
[       C,      1,      0,      1]      -> ^h1f;
[       C,      1,      0,      1]      -> ^h3f;
[       C,      1,      0,      1]      -> ^h7f;
[       C,      1,      1,      1]      -> Z;
[       C,      1,      1,      1]      -> Z;
[       C,      1,      0,      1]      -> ^hFF;
[       C,      0,      0,      1]      -> ^hFE;
[       C,      0,      0,      1]      -> ^hFC;
[       C,      0,      0,      1]      -> ^hF8;
[       C,      0,      0,      1]      -> ^hF0;
[       C,      0,      0,      1]      -> ^hE0;
[       C,      1,      1,      1]      -> Z;
[       0,      1,      0,      1]      -> ^hC1;

end INSREG
```

```
module RXPAR

This module checks the parity of the incoming data with
respect to that calculated. If an error occurs it is held
until the next RXSYNC occurs.

title RECEIVE PARITY CHECKER ATCINT U38

Leo J. Wapelhorst---FAA Tech Center---7-5-90
        U38 device P22V10;


CLK,LDRX                                        pin 1,2;
D0,D1,D2,D3,D4,D5,D6,D7                         pin 3,4,5,6,7,8,9,10;
PARIN,RXSYNC,PLO,PHI,POUT                       pin 11,13,14,15,16;
RXPER,RXMEN,STAT                                pin 17,23,22;

Datain = [D7,D6,D5,D4,D3,D2,D1,D0];
H,L,X,Z,C = 1, 0, .X., .Z., .C.;

equations

PLO = D0 $ D1 $ D2 $ D3;
PHI = D4 $ D5 $ D6 $ D7;
POUT = !(PLO $ PHI);
RXPER := (PARIN $ POUT) & RXSYNC & LDRX & !STAT #
      RXPER & RXSYNC;
!RXMEN = LDRX # STAT;
test_vectors U38 - parity check

([CLK,Datain,PARIN,LDRX,RXSYNC] -> [PHI,PLO,POUT,RXPER]);

[C,0,0,0,1] ->      [0,0,1,0];
[C,^hb1,1,1,1] -> [1,1,1,0];" B1 eom decode
[C,^h31,1,0,1] -> [0,1,0,0];" B1 decode held
[C,^h41,1,0,1] -> [1,1,1,0];
[C,^h81,1,0,1] -> [1,1,1,0];
[C,^h83,1,0,1] -> [1,0,0,0];
[C,^h87,1,0,1] -> [1,1,1,0];
[C,^h8e,1,0,1] -> [1,1,1,0];
[C,^h11,0,0,1] -> [1,1,1,0];Parity error created not enabled
[C,^h11,0,1,1] -> [1,1,1,1];Parity error created
[C,^he1,0,0,1] -> [1,1,1,1];
[C,^h11,0,0,1] -> [1,1,1,1];
[C,^h11,0,0,1] -> [1,1,1,1];
[C,^h33,1,0,0] -> [0,0,1,0];RXSYNC CLRS PARITY ERR

end RXPAR;
```

```
module SYNDET

title RECEIVER SYNC DETECTOR  ATCINT U42

Leo J. Wapelhorst---FAA Tech Center--6-11-90

        U42 device P22V10;

CLK,RXCKEN,RXDVAR,RXEOM,SRESET                  pin 1,2,3,4,11;
STOPRX                                          pin 13;
DISYND,RXSYNC,RXACTIVE,ALLOW                    pin 10,16,18,23;
SIXT,QA,QB,QC,QD                                pin 15,19,20,21,22;

        reset, preset           Node 25,26;

        H,L,X,Z,C = 1, 0, .X., .Z., .C.;
        ST = [QD,QC,QB,QA];
        STA= [RXSYNC,SIXT];

Counter states

S0=^b0000;    S1=^b0001;    S2=^b0011;    S3=^b0010;
S4=^b0110;    S5=^b0111;    S6=^b0101;    S7=^b0100;

S8=^b1100;    S9=^b1101;    S10=^b1111;   S11=^b1110;
S12=^b1010;   S13=^b1011;   S14=^b1001;   S15=^b1000;

STA0=^b11;
STA1=^b10;
STA2=^b00;
STA3=^b01;

equations

![QD,QC,QB,QA] := RXCKEN & RXDVAR # DISYND;

RXACTIVE := RXCKEN & !RXSYNC & !RXACTIVE & !RXEOM & RXDVAR #
        RXACTIVE & !STOPRX;
reset    = !SRESET;

ALLOW := RXDVAR & RXCKEN  #
      ALLOW & !(!RXDVAR & RXCKEN & !RXSYNC & SIXT);
```

```
state_diagram [QD,QC,QB,QA]

State S0:       case RXCKEN & !RXDVAR & ALLOW : S1;
                !RXCKEN:S0;
                endcase;
State S1:       case RXCKEN & !RXDVAR & ALLOW : S2;
                !RXCKEN:S1;
                endcase;
State S2:       case RXCKEN & !RXDVAR & ALLOW : S3;
                !RXCKEN:S2;
                endcase;
State S3:       case RXCKEN & !RXDVAR & ALLOW : S4;
                !RXCKEN:S3;
                endcase;
State S4:       case RXCKEN & !RXDVAR & ALLOW : S5;
                !RXCKEN:S4;
        endcase;
State S5:       case RXCKEN & !RXDVAR & ALLOW : S6;
                !RXCKEN:S5;
                endcase;
State S6:       case RXCKEN & !RXDVAR & ALLOW : S7;
                !RXCKEN:S6;
                endcase;
State S7:       case RXCKEN & !RXDVAR & ALLOW : S8;
                !RXCKEN:S7;
                endcase;
State S8:       case RXCKEN & !RXDVAR & ALLOW : S9;
                !RXCKEN:S8;
                endcase;
State S9:       case RXCKEN & !RXDVAR & ALLOW : S10;
                !RXCKEN:S9;
                endcase;
State S10:case RXCKEN & !RXDVAR & ALLOW : S11;
                !RXCKEN:S10;
                endcase;
State S11:case RXCKEN & !RXDVAR & ALLOW : S12;
                !RXCKEN:S11;
                endcase;
State S12:case RXCKEN & !RXDVAR & ALLOW : S13;
                !RXCKEN:S12;
                endcase;
State S13:case RXCKEN & !RXDVAR & ALLOW : S14;
                !RXCKEN:S13;
                endcase;
State S14:case RXCKEN & !RXDVAR & ALLOW : S15;
                !RXCKEN:S14;
                endcase;
State S15:case RXCKEN & !RXDVAR & ALLOW : S15;
                RXCKEN & !RXDVAR & ALLOW & !RXSYNC & SIXT : S0;
                !RXCKEN:S15;
                        endcase;
```

```
state_diagram [RXSYNC,SIXT]

State STA0:case RXCKEN&!RXDVAR&ALLOW&!QA&!QB&!QC&QD : STA1;
                RXCKEN&RXDVAR :STA0;
                !RXCKEN      :STA0;
                endcase;
State STA1:case RXCKEN&!RXDVAR&ALLOW&!QA&!QB&!QC&QD  : STA2;
                RXCKEN&RXDVAR : STA0;
                !RXCKEN      :STA1;
                endcase;
State STA2:case RXCKEN&!RXDVAR&ALLOW&!QA&!QB&!QC&QD : STA3;
                RXCKEN&RXDVAR : STA0;
                !RXCKEN      :STA2;
                endcase;
State STA3:case RXCKEN&!RXDVAR & ALLOW : STA0;
                RXCKEN&RXDVAR : STA0;
                !RXCKEN      :STA3;
                endcase;
```

test_vectors 17 Zeros followed by a 1

([RXCKEN,CLK,SRESET,RXDVAR,STOPRX,DISYND]->
[ST,SIXT,RXSYNC,RXACTIVE,ALLOW])

```
[1,C,0,0,0,0]    ->    [^hf,1,1,0,0];    RESET
[1,C,1,1,0,0]    ->    [^h0,1,1,0,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,0,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h0,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h1,1,1,0,1];    1ST ZERO
[0,C,1,0,0,0]    ->    [^h1,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h1,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h3,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h3,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h3,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h2,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h2,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h2,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h6,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h6,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h7,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h7,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h7,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h5,1,1,0,1];vector 20
[0,C,1,0,0,0]    ->    [^h5,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h4,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h4,1,1,0,1];
[1,C,1,0,0,0]    ->    [^hc,1,1,0,1];
[0,C,1,0,0,0]    ->    [^hc,1,1,0,1];
[1,C,1,0,0,0]    ->    [^hd,1,1,0,1];
[0,C,1,0,0,0]    ->    [^hd,1,1,0,1];
[0,C,1,0,0,0]    ->    [^hd,1,1,0,1];
[1,C,1,0,0,0]    ->    [^hf,1,1,0,1];
[0,C,1,0,0,0]    ->    [^hf,1,1,0,1];
[1,C,1,0,0,0]    ->    [^he,1,1,0,1];
[0,C,1,0,0,0]    ->    [^he,1,1,0,1];
[1,C,1,0,0,0]    ->    [^ha,1,1,0,1];
[0,C,1,0,0,0]    ->    [^ha,1,1,0,1];
[1,C,1,0,0,0]    ->    [^hb,1,1,0,1];
[0,C,1,0,0,0]    ->    [^hb,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h9,1,1,0,1];
[0,C,1,0,0,0]    ->    [^h9,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h8,1,1,0,1];    15th
[0,C,1,0,0,0]    ->    [^h8,1,1,0,1];    Still there
[0,C,1,0,0,0]    ->    [^h8,1,1,0,1];    vector 41
[1,C,1,0,0,0]    ->    [^h8,0,1,0,1];    16th
[0,C,1,0,0,0]    ->    [^h8,0,1,0,1];
[1,C,1,0,0,0]    ->    [^h8,0,0,0,1];    17th RXSYNC GENERATED
[0,C,1,0,0,0]    ->    [^h8,0,0,0,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];    18th bit = 1,RXSYNC terminated ,RXACTIVE
GENERATED
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
```

test_vectors more than 19 zeros

([RXCKEN,CLK,SRESET,RXDVAR,STOPRX,DISYND]->
[ST,SIXT,RXSYNC,RXACTIVE,ALLOW])

```
[1,C,0,0,0,0]   ->   [^hf,1,1,0,0];   RESET vector 48
[1,C,1,1,0,0]   ->   [^h0,1,1,0,1];
[0,C,1,1,0,0]   ->   [^h0,1,1,0,1];
[1,C,1,1,0,0]   ->   [^h0,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h0,1,1,0,1];   1ST ZERO-no en
[1,C,1,0,0,0]   ->   [^h1,1,1,0,1];   1ST ZERO
[0,C,1,0,0,0]   ->   [^h1,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h3,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h3,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h2,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h2,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h2,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h6,1,1,0,1];vector 60
[0,C,1,0,0,0]   ->   [^h6,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h7,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h7,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h5,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h5,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h4,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h4,1,1,0,1];
[1,C,1,0,0,0]   ->   [^hc,1,1,0,1];
[0,C,1,0,0,0]   ->   [^hc,1,1,0,1];
[1,C,1,0,0,0]   ->   [^hd,1,1,0,1];
[0,C,1,0,0,0]   ->   [^hd,1,1,0,1];
[0,C,1,0,0,0]   ->   [^hd,1,1,0,1];
[1,C,1,0,0,0]   ->   [^hf,1,1,0,1];
[0,C,1,0,0,0]   ->   [^hf,1,1,0,1];
[1,C,1,0,0,0]   ->   [^he,1,1,0,1];
[0,C,1,0,0,0]   ->   [^he,1,1,0,1];
[1,C,1,0,0,0]   ->   [^ha,1,1,0,1];
[0,C,1,0,0,0]   ->   [^ha,1,1,0,1];
[0,C,1,0,0,0]   ->   [^ha,1,1,0,1];
[1,C,1,0,0,0]   ->   [^hb,1,1,0,1];vector 80
[0,C,1,0,0,0]   ->   [^hb,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h9,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h9,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h9,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h8,1,1,0,1];   15th
[0,C,1,0,0,0]   ->   [^h8,1,1,0,1];   15th
[1,C,1,0,0,0]   ->   [^h8,0,1,0,1];   16th vector 40
[0,C,1,0,0,0]   ->   [^h8,0,1,0,1];
[0,C,1,0,0,0]   ->   [^h8,0,1,0,1];
[1,C,1,0,0,0]   ->   [^h8,0,0,0,1];   17th RXSYNC GENERATED
[0,C,1,0,0,0]   ->   [^h8,0,0,0,1];   17th RXSYNC GENERATED
[0,C,1,0,0,0]   ->   [^h8,0,0,0,1];
[0,C,1,0,0,0]   ->   [^h8,0,0,0,1];
[1,C,1,0,0,0]   ->   [^h8,1,0,0,1];   18th bit = 0,RXSYNC generated
[0,C,1,0,0,0]   ->   [^h8,1,0,0,1];
[1,C,1,0,0,0]   ->   [^h0,1,1,0,0];   19th COUNTERS RESET
```

```
[0,C,1,0,0,0]    ->    [^h0,1,1,0,0];
[1,C,1,0,0,0]    ->    [^hf,1,1,0,0];    20th ABORT
[0,C,1,0,0,0]    ->    [^hf,1,1,0,0];    20th ABORT
[0,C,1,0,0,0]    ->    [^hf,1,1,0,0];    vector 100
[1,C,1,0,0,0]    ->    [^hf,1,1,0,0];    21th DOES NOT RESTART
[0,C,1,0,0,0]    ->    [^hf,1,1,0,0];    21th DOES NOT RESTART
[0,C,1,0,0,0]    ->    [^hf,1,1,0,0];    22th DOES NOT RESTART
```

test_vectors restart test - 18 zeros followed by a 1

([RXCKEN,CLK,SRESET,RXDVAR,STOPRX,DISYND]->
[ST,SIXT,RXSYNC,RXACTIVE,ALLOW])

```
[0,C,0,1,0,0]   ->   [^hf,1,1,0,0];vector 104
[0,C,0,1,0,0]   ->   [^hf,1,1,0,0];vector
[1,C,0,1,0,0]   ->   [^hf,1,1,0,0];vector
[1,C,1,1,0,0]   ->   [^h0,1,1,0,1];
[0,C,1,1,0,0]   ->   [^h0,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h1,1,1,0,1];     1ST ZERO
[0,C,1,0,0,0]   ->   [^h1,1,1,0,1];     1ST ZERO
[0,C,1,0,0,0]   ->   [^h1,1,1,0,1];     1ST ZERO
[0,C,1,0,0,0]   ->   [^h1,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h3,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h3,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h2,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h2,1,1,0,1];
[0,C,1,1,0,0]   ->   [^h2,1,1,0,1];     restart - not enab
[1,C,1,1,0,0]   ->   [^h0,1,1,0,1];     restart
[0,C,1,1,0,0]   ->   [^h0,1,1,0,1];     restart
[1,C,1,1,0,0]   ->   [^h0,1,1,0,1];vector 120
[0,C,1,1,0,0]   ->   [^h0,1,1,0,1];
[1,C,1,1,0,0]   ->   [^h0,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h1,1,1,0,1];     1ST ZERO
[0,C,1,0,0,0]   ->   [^h1,1,1,0,1];     1ST ZERO
[1,C,1,0,0,0]   ->   [^h3,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h3,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h2,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h2,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h2,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h6,1,1,0,1];VECTOR 130
[0,C,1,0,0,0]   ->   [^h6,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h7,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h7,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h5,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h5,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h4,1,1,0,1];
[0,C,1,0,0,0]   ->   [^h4,1,1,0,1];
[1,C,1,0,0,0]   ->   [^hc,1,1,0,1];
[0,C,1,0,0,0]   ->   [^hc,1,1,0,1];
[0,C,1,0,0,0]   ->   [^hc,1,1,0,1];
[1,C,1,0,0,0]   ->   [^hd,1,1,0,1];
[0,C,1,0,0,0]   ->   [^hd,1,1,0,1];
[1,C,1,0,0,0]   ->   [^hf,1,1,0,1];
[0,C,1,0,0,0]   ->   [^hf,1,1,0,1];
[1,C,1,0,0,0]   ->   [^he,1,1,0,1];
[0,C,1,0,0,0]   ->   [^he,1,1,0,1];
[1,C,1,0,0,0]   ->   [^ha,1,1,0,1];
[0,C,1,0,0,0]   ->   [^ha,1,1,0,1];
[0,C,1,0,0,0]   ->   [^ha,1,1,0,1];
[1,C,1,0,0,0]   ->   [^hb,1,1,0,1];VECTOR 150
[0,C,1,0,0,0]   ->   [^hb,1,1,0,1];
[1,C,1,0,0,0]   ->   [^h9,1,1,0,1];
```

```
[0,C,1,0,0,0]    ->    [^h9,1,1,0,1];
[1,C,1,0,0,0]    ->    [^h8,1,1,0,1];    15th
[0,C,1,0,0,0]    ->    [^h8,1,1,0,1];    15th
[0,C,1,0,0,0]    ->    [^h8,1,1,0,1];    15th
[1,C,1,0,0,0]    ->    [^h8,0,1,0,1];    16th
[0,C,1,0,0,0]    ->    [^h8,0,1,0,1];    16th
[1,C,1,0,0,0]    ->    [^h8,0,0,0,1];    17th RXSYNC GENERATED
[0,C,1,0,0,0]    ->    [^h8,0,0,0,1];
[1,C,1,0,0,0]    ->    [^h8,1,0,0,1];    18th RXSYNC GENERATED,RXACTIVE NOT
GENERATED (data=0)
[0,C,1,0,0,0]    ->    [^h8,1,0,0,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];    vector 73 RXACTIVE GENERATED
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];    V165
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,0,0,0]    ->    [^h1,1,1,1,1];
[0,C,1,0,0,0]    ->    [^h1,1,1,1,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];    VECTOR 170
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,0,0,0]    ->    [^h1,1,1,1,1];
[0,C,1,0,0,0]    ->    [^h1,1,1,1,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,0,0,0]    ->    [^h1,1,1,1,1];
[0,C,1,0,0,0]    ->    [^h1,1,1,1,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[0,C,1,1,0,0]    ->    [^h0,1,1,1,1];
[1,C,1,0,0,0]    ->    [^h1,1,1,1,1];    VECTOR 190
[0,C,1,0,0,0]    ->    [^h1,1,1,1,1];
[1,C,1,0,1,0]    ->    [^h3,1,1,0,1];    STOPRX,RXACTIVE TERMINATED
[0,C,1,0,1,0]    ->    [^h3,1,1,0,1];
[1,C,1,1,0,1]    ->    [^h0,1,1,0,1];
[0,C,1,1,0,1]    ->    [^h0,1,1,0,1];    VECTOR 195

end SYNDET;
```

```
module BITCOUNT

title RECEIVER BIT COUNTER  ATCINT-2 U43
Leo J. Wapelhorst---FAA Tech Center-----6-27-90

        U43 device P22V10;

This Pal contains the logic to control storage of receive
data in memory. When the consecutive zero threshold is
exceeded (MTE), incoming date is stored in Ram as it comes
in (every 8 bits). If sync is detected, the address
counter is reset and data is stored every 9 bits as parity
is not stored in memory for real data.

This is accomplished in the following manner: A state
controller is in the IDLE state until MTE occurs. When
this happens, the state controller goes around a loop as
follows:S2-S3-S4-S5-S6-S7-S8-S9-S2-S3 etc. Data is
written into Ram on S9. If sync occurs, the state
controller goes to state S1 and also loops back to S1
instead of S2 making the loop equal to nine clocks. Data
is still stored at state S9.

CLK,RXCKEN,RXSYNC,RXACTIVE,!RESET          pin 1,2,3,4,11;
MTE,STOPRX                                 pin 5,10;
LOADRX,LACT,RES                            pin 18,17,14;
QA,QB,QC,QD                                pin 20,21,22,23;

                                           reset,preset          node
25,26;

        H,L,X,Z,C = 1, 0, .X., .Z., .C.;
        ST = [QD,QC,QB,QA];

Counter states

S0 = ^b1111;    S1 = ^b1110;    S2 = ^b1010;    S3 = ^b1011;
S4 = ^b1001;    S5 = ^b1000;    S6 = ^b0000;    S7 = ^b0001;
S8 = ^b0011;    S9 = ^b0111;

equations

RES       = RESET # !RXSYNC # STOPRX;
reset     = RES;

LOADRX    = !QD & QC & QB & QA;
LACT      := MTE & RXCKEN # LACT & RXSYNC;
```

```
state_diagram [QD,QC,QB,QA]

State S0:       case RXCKEN & RXSYNC & RXACTIVE :S1;
                     RXCKEN & MTE & !LACT       :S2;
                     RXCKEN & !RXACTIVE         :S0;
        !RXCKEN :S0;
                endcase;
State S1:       case RXCKEN                     : S2;
        !RXCKEN                 : S1;
                endcase;
State S2:       case RXCKEN : S3;
        !RXCKEN : S2;
                endcase;
State S3:       case RXCKEN : S4;
        !RXCKEN : S3;
                endcase;
State S4:       case RXCKEN : S5;
        !RXCKEN : S4;
                endcase;
State S5:       case RXCKEN : S6;
        !RXCKEN : S5;
                endcase;
State S6:       case RXCKEN : S7;
        !RXCKEN : S6;
                endcase;
State S7:       case RXCKEN : S8;
        !RXCKEN : S7;
                endcase;
State S8:       case RXCKEN : S9;
        !RXCKEN : S8;
                endcase;
State S9:       case RXCKEN &  RXSYNC & RXACTIVE : S1;
         RXCKEN & LACT          : S2;
        !RXCKEN  : S9;
                endcase;
```

```
test_vectors

([RXCKEN,CLK,!RESET,RXSYNC,RXACTIVE,MTE,STOPRX]-> [ST,LOADRX,LACT])

[1,0,0,1,0,0,0] -> [^hf,0,0];          RESET
[1,C,1,1,0,0,0] -> [^hf,0,0];
[0,C,1,1,0,0,0] -> [^hf,0,0];
[0,C,1,1,0,0,0] -> [^hf,0,0];
[1,C,1,0,0,0,0] -> [^hf,0,0];          RXSYNC
[0,C,1,0,0,0,0] -> [^hf,0,0];          RXSYNC
[0,C,1,1,1,0,0] -> [^hf,0,0];          RXACTIVE,-NOT ENAB
[1,C,1,1,1,0,0] -> [^he,0,0];          RXACTIVE,RPT GEN
[0,C,1,1,1,0,0] -> [^he,0,0];          RXACTIVE,RPT GEN
[0,C,1,1,1,0,0] -> [^he,0,0];          RXACTIVE,RPT GEN
[1,C,1,1,1,0,0] -> [^ha,0,0];
[0,C,1,1,1,0,0] -> [^ha,0,0];
[1,C,1,1,1,0,0] -> [^hb,0,0];
[0,C,1,1,1,0,0] -> [^hb,0,0];
[1,C,1,1,1,0,0] -> [^h9,0,0];
[0,C,1,1,1,0,0] -> [^h9,0,0];
[1,C,1,1,1,0,0] -> [^h8,0,0];
[0,C,1,1,1,0,0] -> [^h8,0,0];
[1,C,1,1,1,0,0] -> [^h0,0,0];
[0,C,1,1,1,0,0] -> [^h0,0,0];VECTOR 20
[0,C,1,1,1,0,0] -> [^h0,0,0];
[1,C,1,1,1,0,0] -> [^h1,0,0];
[0,C,1,1,1,0,0] -> [^h1,0,0];
[1,C,1,1,1,0,0] -> [^h3,0,0];
[0,C,1,1,1,0,0] -> [^h3,0,0];
[1,C,1,1,1,0,0] -> [^h7,1,0];          LOADRX GEN
[0,C,1,1,1,0,0] -> [^h7,1,0];          LOADRX GEN
[1,C,1,1,1,0,0] -> [^hc,0,0];
[0,C,1,1,1,0,0] -> [^hc,0,0];
[1,C,1,1,1,0,0] -> [^ha,0,0];
[0,C,1,1,1,0,0] -> [^ha,0,0];
[1,C,1,1,1,0,0] -> [^hb,0,0];
[0,C,1,1,1,0,0] -> [^hb,0,0];
[1,C,1,1,1,0,0] -> [^h9,0,0];
[0,C,1,1,1,0,0] -> [^h9,0,0];
[1,C,1,1,1,0,0] -> [^h8,0,0];
[0,C,1,1,1,0,0] -> [^h8,0,0];
[1,C,1,1,1,0,0] -> [^h0,0,0];
[0,C,1,1,1,0,0] -> [^h0,0,0];
[1,C,1,1,1,0,0] -> [^h1,0,0];VECTOR 40
[0,C,1,1,1,0,0] -> [^h1,0,0];
[1,C,1,1,1,0,0] -> [^h3,0,0];
[0,C,1,1,1,0,0] -> [^h3,0,0];
[1,C,1,1,1,0,0] -> [^h7,1,0];          LOADRX GEN
[0,C,1,1,1,0,0] -> [^h7,1,0];          LOADRX GEN
[1,C,1,1,1,0,0] -> [^hc,0,0];
[0,C,1,1,1,0,0] -> [^hc,0,0];
[1,C,1,1,0,0,0] -> [^ha,0,0];          END OF RXACTIVE
[0,C,1,1,0,0,0] -> [^ha,0,0];          END OF RXACTIVE
[1,C,1,1,0,0,0] -> [^hb,0,0];VECTOR 50
```

```
[0,C,1,1,0,0,0] -> [^hb,0,0];
[X,C,1,1,0,0,1] ->          [^hf,0,0];
[1,C,1,1,0,1,0] -> [^ha,0,1];LACT (MTE)-S2
[0,C,1,1,0,1,0] -> [^ha,0,1];
[1,C,1,1,0,0,0] -> [^hb,0,1];S3
[0,C,1,1,0,0,0] -> [^hb,0,1];
[1,C,1,1,0,0,0] -> [^h9,0,1];S4
[0,C,1,1,0,0,0] -> [^h9,0,1];
[1,C,1,1,0,0,0] -> [^h8,0,1];S5
[0,C,1,1,0,0,0] -> [^h8,0,1];
[1,C,1,1,0,0,0] -> [^h0,0,1];S6 - vector 60
[0,C,1,1,0,0,0] -> [^h0,0,1];
[1,C,1,1,0,0,0] -> [^h1,0,1];S7
[0,C,1,1,0,0,0] -> [^h1,0,1];
[1,C,1,1,0,0,0] -> [^h3,0,1];S8
[0,C,1,1,0,0,0] -> [^h3,0,1];
[1,C,1,1,0,0,0] -> [^h7,1,1];S9 - LOADRX
[0,C,1,1,0,0,0] -> [^h7,1,1]; LOADRX
[1,C,1,1,0,0,0] -> [^ha,0,1];S2
[0,C,1,1,0,0,0] -> [^ha,0,1];
[1,C,1,0,0,0,0] -> [^hf,0,0];RXSYNC
[0,C,1,0,0,0,0] -> [^hf,0,0];RXSYNC
[0,C,1,1,1,0,0] -> [^hf,0,0];          RXACTIVE,-NOT ENAB
[1,C,1,1,1,0,0] -> [^he,0,0];          RXACTIVE,RPT GEN
[0,C,1,1,1,0,0] -> [^he,0,0];          RXACTIVE,RPT GEN
[0,C,1,1,1,0,0] -> [^he,0,0];          RXACTIVE,RPT GEN
[1,C,1,1,1,0,0] -> [^ha,0,0];
[0,C,1,1,1,0,0] -> [^ha,0,0];
[1,C,1,1,1,0,0] -> [^hb,0,0];
[0,C,1,1,1,0,0] -> [^hb,0,0];
[1,C,1,1,1,0,0] -> [^h9,0,0];
[0,C,1,1,1,0,0] -> [^h9,0,0];
[1,C,1,1,1,0,0] -> [^h8,0,0];
[0,C,1,1,1,0,0] -> [^h8,0,0];
[1,C,1,1,1,0,0] -> [^h0,0,0];

end BITCOUNT;
```

```
module EOMDET

title RECEIVE END OF MESSAGE DETECTOR  ATCINT U44

Leo J. Wapelhorst---FAA Tech Center---7-13-90
        U44 device P22V10;

CLK,LDRX                                                pin 1,2;
D0,D1,D2,D3,D4,D5,D6,D7                                 pin 3,4,5,6,7,8,9,10;
PARIN,RXACTIVE,RXPER,LDRX1                              pin 11,13,19,18;
LRCPREP,RXEOM,ABORT,DISEOM                              pin 23,22,21,20;
STOPRX,Res_Eom                                          pin 14,15;

Datain = [D7,D6,D5,D4,D3,D2,D1,D0];
H,L,X,Z,C = 1, 0, .X., .Z., .C.;

equations

RXEOM  := (LDRX & RXACTIVE &
       (((Datain==^hB1) & !DISEOM) # RXPER)) #
       RXEOM & !STOPRX & !Res_Eom;
LRCPREP := LDRX & !LDRX1 & (Datain==^hB3) #
       LRCPREP & !(LDRX & !LDRX1);
DISEOM := LRCPREP & !LDRX & LDRX1      #
       DISEOM & !(!LDRX & LDRX1);
LDRX1  := LDRX;
ABORT  := (Datain==^h00) & !PARIN;
RXEOM.re = !RXACTIVE;
ABORT.re = !RXACTIVE;
DISEOM.re= !RXACTIVE;
LRCPREP.re=!RXACTIVE;

test_vectors
([CLK,Datain,PARIN,LDRX,RXACTIVE] -> [LRCPREP,RXEOM,ABORT,DISEOM]);

[C,0,0,0,0]       ->      [0,0,0,0];
[C,^hb0,1,1,1]    ->      [0,0,0,0];
[C,^hb0,0,0,1]    ->      [0,0,0,0];
[C,^hb3,1,1,1]    ->      [1,0,0,0];LRCPREP
[C,^hb3,1,1,1]    ->      [1,0,0,0];
[C,^hb3,0,0,1]    ->      [1,0,0,1];
[C,^hb4,0,0,1]    ->      [1,0,0,1];
[C,^hb2,0,0,1]    ->      [1,0,0,1];
[C,^hb1,1,1,1]    ->      [0,0,0,1];B1 DECODE - BLOCKED
[C,^hb1,1,1,1]    ->      [0,0,0,1];B1 DECODE - BLOCKED
[C,^hb1,1,1,1]    ->      [0,0,0,1];B1 DECODE - BLOCKED
[C,^hb1,0,0,1]    ->      [0,0,0,0];DISEOM ENABLED
[C,^hb0,1,1,1]    ->      [0,0,0,0];
[C,^hb0,0,0,1]    ->      [0,0,0,0];
[C,^hb3,1,1,1]    ->      [1,0,0,0];LRCPREP
[C,^hb3,1,1,1]    ->      [1,0,0,0];
[C,^hb3,0,0,1]    ->      [1,0,0,1];
[C,^hb3,0,0,0]    ->      [0,0,0,0];
end EOMDET;
```

```
module ZEROMON

title RECEIVER CONSECUTIVE ZERO MONITOR  ATCINT-2 U45
        Leo J. Wapelhorst---FAA Tech Center-----7-19-90

U45 device P22V10;

This Pal contains the counter which monitors the number
of consecutive zeros which occur in the incoming data
 When the consecutive zero threshold is
exceeded (MTE), incoming date is stored in Ram as it comes
in (every 8 bits).  If sync is detected, the address
counter is reset and data is stored every 9 bits as parity
is not stored in memory for real data.

The counter is reset whenever a data bit = 1 or when RXEOM is
received.

CLK,RXCKEN,RXDVAR,RXEOM,SRESET                  pin 1,2,3,4,11;
MON0,MON1,MON2,MON3,MON4                        pin 5,6,7,8,9;
RXACTIVE                                        pin 10;
MENAB,MTE,EN                                    pin 23,17,16;
M0,M1,M2,M3,M4                                  pin 18,19,20,21,22;

reset,preset            node 25,26;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
MIN     = [MON4,MON3,MON2,MON1,MON0];
MOUT    = [M4,M3,M2,M1,M0];
MOUT_re  = [M4.re,M3.re,M2.re,M1.rc,M0.re];

equations

MOUT_re  = !SRESET;
MOUT:=!RXEOM & !RXDVAR & RXCKEN & (MOUT+1) & EN #
    MOUT & !RXCKEN;
MENAB = (M4==MON4) & (M3==MON3) & (M2==MON2);
MTE  = (M1==MON1) & (M0==MON0) & MENAB;
EN  := RXDVAR & RXCKEN & !EN & !RXACTIVE #
     EN & !MTE & !RXACTIVE;

test_vectors

([RXCKEN,CLK,SRESET,RXEOM,RXDVAR,MIN,RXACTIVE]-> [MOUT,MENAB,MTE,EN])

[1,0,0,0,0,^h5,0] ->      [^h0,0,0,0];      RESET
[1,C,1,0,0,^h5,0] ->      [^h0,0,0,0];
[0,C,1,0,0,^h5,0] ->      [^h0,0,0,0];
[0,C,1,0,0,^h5,0] ->      [^h0,0,0,0];
[1,C,1,0,1,^h5,0] ->      [^h0,0,0,1];ENABLED BY A 1
[0,C,1,0,0,^h5,0] ->      [^h0,0,0,1];
[0,C,1,0,0,^h5,0] ->      [^h0,0,0,1];
[1,C,1,0,0,^h5,0] ->      [^h1,0,0,1];
[0,C,1,0,0,^h5,0] ->      [^h1,0,0,1];
```

```
[0,C,1,0,0,^h5,0] ->    [^h1,0,0,1];
[1,C,1,0,0,^h5,0] ->    [^h2,0,0,1];
[0,C,1,0,0,^h5,0] ->    [^h2,0,0,1];
[0,C,1,0,1,^h5,0] ->    [^h2,0,0,1];        RESET DATA = 1
[1,C,1,0,1,^h5,0] ->    [^h0,0,0,1];
[0,C,1,0,1,^h5,0] ->    [^h0,0,0,1];
[0,C,1,0,1,^h5,0] ->    [^h0,0,0,1];
[1,C,1,0,1,^h5,0] ->    [^h0,0,0,1];
[0,C,1,0,1,^h5,0] ->    [^h0,0,0,1];
[1,C,1,0,0,^h5,0] ->    [^h1,0,0,1];
[0,C,1,0,0,^h5,0] ->    [^h1,0,0,1];
[1,C,1,0,0,^h5,0] ->    [^h2,0,0,1];
[0,C,1,0,0,^h5,0] ->    [^h2,0,0,1];
[1,C,1,0,0,^h5,0] ->    [^h3,0,0,1];
[0,C,1,0,0,^h5,0] ->    [^h3,0,0,1];
[1,C,1,0,0,^h5,0] ->    [^h4,1,0,1];
[0,C,1,0,0,^h5,0] ->    [^h4,1,0,1];
[0,C,1,0,0,^h5,0] ->    [^h4,1,0,1];
[1,C,1,0,0,^h5,0] ->    [^h5,1,1,1];MTE ISSUED
[0,C,1,0,0,^h5,0] ->    [^h5,1,1,0];
[1,C,1,0,0,^h5,0] ->    [^h0,0,0,0];
[0,C,1,0,0,^h5,0] ->    [^h0,0,0,0];
[1,C,1,0,0,^h5,0] ->    [^h0,0,0,0];
[0,C,1,0,0,^h5,0] ->    [^h0,0,0,0];
[1,C,0,0,0,^h5,0] ->    [^h0,0,0,0];
[1,C,1,0,0,^h5,0] ->    [^h0,0,0,0];
[0,C,1,0,0,^h5,1] ->    [^h0,0,0,0];SYNC DETECTED
[1,C,1,0,0,^h5,1] ->    [^h0,0,0,0];
[0,C,1,0,0,^h5,1] ->    [^h0,0,0,0];
[1,C,1,0,1,^h5,1] ->    [^h0,0,0,0];1-NOT ENABLED
[1,C,1,0,1,^h5,1] ->    [^h0,0,0,0];1-NOT ENABLED
[1,C,1,0,1,^h5,1] ->    [^h0,0,0,0];1-NOT ENABLED
[1,C,1,0,0,^h5,1] ->    [^h0,0,0,0];0-NOT ENABLED
[1,C,1,0,0,^h5,1] ->    [^h0,0,0,0];0-NOT ENABLED
[1,C,1,0,0,^h5,1] ->    [^h0,0,0,0];0-NOT ENABLED
[1,C,1,0,0,^h5,1] ->    [^h0,0,0,0];0-NOT ENABLED
[1,C,1,0,0,^h5,1] ->    [^h0,0,0,0];0-NOT ENABLED
[1,C,1,0,0,^h5,1] ->    [^h0,0,0,0];0-NOT ENABLED

end ZEROMON;
```

```
module STATCONA

title STATUS CONTROL LOGIC-- ATC INTERFACE  U46A,47

Leo J. Wapelhorst---FAA Tech Center---7-17-90
   U46A,U47  device P22V10;

CLK,RXCKEN,RXEOM,LDRX                    pin in U46A 1,2,3,4;
RXPER,RXACTIVE,ACAR                      pin in U46A 5,6,7;
SRESET,RXBUSY                            pin in U46A 11,8;
ST0,ST1,ST2,ST3,ST4                      pin in U46A 21,20,19,18,17;
LDRX1,LDRX2,RXACTD                       pin in U46A 16,15,23;
RXINT,STOPRX                             pin in U46A 14,22;

reset,preset                             node in U46A 25,26;

CLK,ST0,ST1,ST2,ST3                      pin in U47 1,3,4,5,6;
ST4,SRESET                               pin in U47 7,13;
En_Rmac,Clr_Rmac,Lo_Rmac                 pin in U47 23,22,21;
STAT,NEXT,Latlad                         pin in U47 20,19,18;
Sel_S1,Sel_S0,RXWR                       pin in U47 17,16,15;
Res_Eom                                  pin in U47 14;
reset,preset                             node in U47 25,26;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
STA    = [ST4,ST3,ST2,ST1,ST0];
STA_re = [ST4.re,ST3.re,ST2.re,ST1.re,ST0.re];
ST4 ISTYPE  in U46A POS;
ST3 ISTYPE  in U46A POS;
ST2 ISTYPE  in U46A POS;
ST1 ISTYPE  in U46A POS;
ST0 ISTYPE  in U46A POS;
```

Output states in U46A

```
S0=^b00000;   S1=^b00001;   S2=^b00010;   S3=^b00011;
S4=^b00100;   S5=^b00101;   S6=^b00110;   S7=^b00111;
S8=^b01000;   S9=^b01001;   SA=^b01010;   SB=^b01011;
SC=^b01100;   SD=^b01101;   SE=^b01110;   SF=^b01111;
S10=^b10000;  S11=^b10001;  S12=^b10010;  S13=^b10011;
S14=^b10100;  S15=^b10101;  S16=^b10110;
```

state_diagram in U46A [ST4,ST3,ST2,ST1,ST0]

```
        State S0:case     !RXACTIVE & LDRX1 & !LDRX2   :S1;
           LDRX1 & !LDRX2 & RXACTIVE & !RXPER  :S1;
           LDRX1 & !LDRX2 & RXACTIVE & RXPER   :S3;
           LDRX1 & !LDRX2 & RXACTIVE & RXEOM   :S3;
           LDRX1 & !LDRX2 & ACAR               :S3;
           RXACTIVE & !RXACTD                  :SE;
         endcase;
        State S1:GOTO S2;
        State S2:GOTO S0;
        State S3:GOTO S4;
        State S4:GOTO S5;
        State S5:GOTO S6;
        State S6:GOTO S7;
        State S7:GOTO S8;
        State S8:GOTO S9;
        State S9:GOTO SA;
        State SA:GOTO SB;
        State SB:case RXACTIVE & RXPER # !RXACTIVE      :SD;
           RXACTIVE & !RXPER & !RXEOM       :SC;
           RXACTIVE & !RXPER & RXEOM        :S10;
         endcase;
        State SC:GOTO S14;SKIP INTERRUPT
        State SD:GOTO SF;
        State SE:GOTO S0;
        State SF:GOTO S14;
        State S10:GOTO S11;
        State S11:GOTO S13;CHANGED TO STOP ON B1
        State S12:case LDRX1 & !LDRX2 & RXACTIVE & !RXPER :S16;
           LDRX1 & !LDRX2 & RXACTIVE & RXPER   :S13;
           !(LDRX1 & !LDRX2)                   :S12;
           endcase;
        State S13:GOTO S0;
        State S14:GOTO S0;
        State S16:GOTO S1;
```

equations in U46A

```
STA_re  = !SRESET;
RXINT  := (STA==^hF) # (STA==^h10) ;
LDRX1  := LDRX;
LDRX2  := LDRX1;
RXACTD := RXACTIVE;
STOPRX  = (STA==^hD) # (STA==^h13);
```

equations in U47

```
!RXWR  := (STA==^h1) # (STA==^h3) # (STA==^h7) # (STA==^hA);
En_Rmac  := (STA==^h2) # (STA==^h8);
Latlad  := (STA==^h4);
!Lo_Rmac := (STA==^h5);
STAT    := (STA==^h5) & !STAT # STAT & !(STA==^hB);5 to C
!Sel_S0  := (STA==^h5)&Sel_S0  # !Sel_S0&!(STA==^h8);5 to 8
!Sel_S1  := (STA==^h9)&Sel_S1  # !Sel_S1&!(STA==^hB);9 to C
!Clr_Rmac := (STA==^hB) # (STA==^hE);
NEXT    := (STA==^h14) # (STA==^h11);
Res_Eom  := (STA==^h16);
```

test_vectors in U46A

```
([SRESET,CLK,RXCKEN,RXEOM,LDRX,RXPER,ACAR,RXACTIVE] ->
[STA,RXINT,LDRX1,LDRX2])

[0,C,0,0,0,0,0,0]  ->     [^h0,0,0,0];     RESET
[1,C,0,0,0,0,0,0]  ->     [^h0,0,0,0];     RESET
[1,C,1,0,0,0,0,1]  ->     [^he,0,0,0];     clear Rmadc
[1,C,1,0,0,0,0,1]  ->     [^h0,0,0,0];
[1,C,1,0,0,0,0,1]  ->     [^h0,0,0,0];
[1,C,1,0,0,0,0,1]  ->     [^h0,0,0,0];
[1,C,1,0,1,0,0,1]  ->     [^h0,0,1,0];     state 1 normal wr
[1,C,1,0,1,0,0,1]  ->     [^h1,0,1,1];     state 1 normal wr
[1,C,X,0,1,0,0,1]  ->     [^h2,0,1,1];     state 2
[1,C,X,0,1,0,0,1]  ->     [^h0,0,1,1];     state 0
[1,C,X,0,1,0,0,1]  ->     [^h0,0,1,1];
[1,C,X,0,1,0,0,1]  ->     [^h0,0,1,1];
NORMAL END OF MESSAGE TEST
[1,C,X,0,0,0,0,1]  ->     [^h0,0,0,1];     VECT 13
[1,C,X,0,0,0,0,1]  ->     [^h0,0,0,0];
[1,C,X,0,0,0,0,1]  ->     [^h0,0,0,0];     state 0
[1,C,1,1,1,0,0,1]  ->     [^h0,0,1,0];     state 0
[1,C,1,1,1,0,0,1]  ->     [^h3,0,1,1];     state 3
[1,C,X,1,1,0,0,1]  ->     [^h4,0,1,1];     state 4
[1,C,1,1,1,0,0,1]  ->     [^h5,0,1,1];     state 5
[1,C,1,1,1,0,0,1]  ->     [^h6,0,1,1];     state 6
[1,C,1,1,1,0,0,1]  ->     [^h7,0,1,1];     state 7
[1,C,1,1,1,0,0,1]  ->     [^h8,0,1,1];     state 8
[1,C,1,1,1,0,0,1]  ->     [^h9,0,1,1];     state 9
[1,C,1,1,1,0,0,1]  ->     [^hA,0,1,1];     state A
[1,C,1,1,1,0,0,1]  ->     [^hB,0,1,1];     state C
[1,C,1,1,1,0,0,1]  ->     [^h10,0,1,1];    state 10
```

```
[1,C,1,1,1,0,0,1]  ->    [^h11,1,1,1];    state 0-RXINT
[1,C,1,1,0,0,0,1]  ->    [^h12,0,0,1];
[1,C,0,1,0,0,0,1]  ->    [^h12,0,0,0];
[1,C,0,1,0,0,0,1]  ->    [^h12,0,0,0];
[1,C,1,1,1,1,0,1]  ->    [^h12,0,1,0];
[1,C,1,1,1,1,0,1]  ->    [^h13,0,1,1];    state 13-STOP
[1,C,1,1,1,1,0,1]  ->    [^h0,0,1,1];     state 0-
[1,C,1,1,1,1,0,1]  ->    [^h0,0,1,1];     state 0-
[1,C,1,1,1,1,0,1]  ->    [^h0,0,1,1];     state 0-
RECEIVE PARITY ERROR WHILE ACTIVE
[1,C,X,0,0,0,0,1]  ->    [^h0,0,0,1];     VECTOR 36
[1,C,X,0,0,1,0,1]  ->    [^h0,0,0,0];
[1,C,X,0,0,1,0,1]  ->    [^h0,0,0,0];     state 0
[1,C,1,X,1,1,0,1]  ->    [^h0,0,1,0];     state 0
[1,C,1,X,1,1,0,1]  ->    [^h3,0,1,1];     state 3
[1,C,X,1,1,1,0,1]  ->    [^h4,0,1,1];     state 4
[1,C,1,X,1,1,0,1]  ->    [^h5,0,1,1];     state 5
[1,C,1,X,1,1,0,1]  ->    [^h6,0,1,1];     state 6
[1,C,1,X,1,1,0,1]  ->    [^h7,0,1,1];     state 7
[1,C,1,X,1,1,0,1]  ->    [^h8,0,1,1];     state 8
[1,C,1,X,1,1,0,1]  ->    [^h9,0,1,1];     state 9
[1,C,1,X,1,1,0,1]  ->    [^hA,0,1,1];     state A
[1,C,1,X,1,1,0,1]  ->    [^hB,0,1,1];     state B
[1,C,1,X,1,1,0,1]  ->    [^hD,0,1,1];     state D
[1,C,1,X,1,1,0,1]  ->    [^hF,0,1,1];     state F-V50
[1,C,1,X,1,1,0,1]  ->    [^h14,1,1,1];    state 14-RXINT
[1,C,1,X,1,1,0,1]  ->    [^h0,0,1,1];     state 0-
[1,C,1,X,1,1,0,1]  ->    [^h0,0,1,1];     state 0-
[1,C,1,X,1,1,0,1]  ->    [^h0,0,1,1];     state 0-
[1,C,1,X,1,1,0,1]  ->    [^h0,0,1,1];     state 0-
```

```
test_vectors in U47
([SRESET,CLK,STA] ->
[En_Rmac,Clr_Rmac,Lo_Rmac,STAT,NEXT,Latlad,Sel_S0,Sel_S1,RXWR])
[0,C,^h0]        ->     [0,1,1,0,0,0,1,1,1];RESET
[1,C,^h0]        ->     [0,1,1,0,0,0,1,1,1];IDLE
[1,C,^h1]        ->     [0,1,1,0,0,0,1,1,0];WRITE
[1,C,^h2]        ->     [1,1,1,0,0,0,1,1,1];EN AD COUNT
[1,C,^h0]        ->     [0,1,1,0,0,0,1,1,1];IDLE
[1,C,^h3]        ->     [0,1,1,0,0,0,1,1,0];WRITE
[1,C,^h4]        ->     [0,1,1,0,0.1,1,1,1];LATCH LAST AD
[1,C,^h5]        ->     [0,1,0,1,0,0,0,1,1];LOAD STAT TO RMAC
[1,C,^h6]        ->     [0,1,1,1,0,0,0,1,1];
[1,C,^h7]        ->     [0,1,1,1,0,0,0,1,0];WRITE S0
[1,C,^h8]        ->     [1,1,1,1,0,0,1,1,1];EN AD CTR
[1,C,^h9]        ->     [0,1,1,1,0,0,1,0,1];SEL S1
[1,C,^hA]        ->     [0,1,1,1,0,0,1,0,0];WR S1
[1,C,^hB]        ->     [0,0,1,0,0,0,1,1,1];RESET STAT,CLR LSB
[1,C,^hD]        ->     [0,1,1,0,0,0,1,1,1];
[1,C,^hF]        ->     [0,1,1,0,0,0,1,1,1];
[1,C,^h14]->    [0,1,1,0,1,0,1,1,1];NEXT
[1,C,^h0]        ->     [0,1,1,0,0,0,1,1,1];IDLE
[1,C,^hE]        ->     [0,0,1,0,0,0,1,1,1];LE RXACTIVE
[1,C,^h0]        ->     [0,1,1,0,0,0,1,1,1];IDLE
end STATCONA
```

```
module RADCON

title RECEIVE MEMORY ADDRESS CONTROL LOGIC -- ATCINT U49,50
Leo J. Wapelhorst---FAA Tech Center-----7-5-90
   U49,U50 device E0600;

This module contains the memory address control for the
receive data memory. It consists of a 7 bit counter for
the Lsb's and a 3 bit counter for the Msb's. The Msb
counter incrementers on a carry from the Lsb or NEXT from the
status control logic. When the STATUS signal occurs, the
Lsb's are loaded to the value corresponding to the status
word for the data group which is stored. The 2 Lsb's of
the 7 bit counter  and all three Msb's are loaded to 0
so that the status is written in to the base address of the
memory. When status is over the address lines are again
set to equal the counter value until incremented by the
status control logic.

CLK,CLK1,En_Rmac,Clr_Rmac,Lo_Rmac          pin in U49 1,13,2,11,14;
D2,D3,D4,STAT                              pin in U49 3,4,5,23;
Q3,Q2,Q1,Q0                                pin in U49 18,17,16,15;
Q4,Q5,Q6                                   pin in U49 19,20,21;
CAR                                        pin in U49 22;
Ad5,Ad6                                    pin in U49 6,7;

CLK,CLK1,En_Rmac,SRESET,Lo_Rmac            pin in U50 1,13,2,11,14;
NEXT,RSYN,STAT                             pin in U50 3,4,23;
Q7,Q8,Q9                                   pin in U50 15,16,17;
Ad7,Ad8,Ad9                                pin in U50 18,19,20;
CAR                                        pin in U50 22;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
DIN   = [D4,D3,D2];
COUNT  = [Q4,Q3,Q2,Q1,Q0];
MCNT   = [Q6,Q5];
MSBCT  = [Q9,Q8,Q7];
MAD    = [Ad9,Ad8,Ad7];
COUNT_re = [Q4.re,Q3.re,Q2.re,Q1.re,Q0.re];
MCNT_re  = [Q6.re,Q5.re];
```

equations IN U49

```
COUNT_re = !Clr_Rmac;
MCNT_re  = !Clr_Rmac;
CAR    = (COUNT = = ^h1F) & Q5 & Q6;
COUNT  := (((COUNT+1)&En_Rmac)#COUNT&!En_Rmac)&Lo_Rmac;
MCNT  := (((MCNT+1)&En_Rmac&(COUNT= =^h1F)) #
        MCNT&!En_Rmac            #
        MCNT & En_Rmac & !(COUNT= =^h1F));
Ad5    = Q5 & !STAT;
Ad6    = Q6 & !STAT;

WHEN !Lo_Rmac THEN Q0 := L;
WHEN !Lo_Rmac THEN Q1 := L;
WHEN !Lo_Rmac THEN Q2 := D2;
WHEN !Lo_Rmac THEN Q3 := D3;
WHEN !Lo_Rmac THEN Q4 := D4;
```

equations IN U50

```
MSBCT  := !RSYN & (((MSBCT+1) & En_Rmac & CAR)      #
       (MSBCT+1) & NEXT            #
        MSBCT & !En_Rmac & !NEXT     #
        MSBCT & En_Rmac & !NEXT & !CAR);
Ad7    = Q7 & !STAT;
Ad8    = Q8 & !STAT;
Ad9    = Q9 & !STAT;
RSYN   = !SRESET # ((MSBCT= =^h7) & NEXT);

WHEN RSYN THEN Q7 := H;SRESET PRESETS COUNT TO 1
WHEN RSYN THEN Q8 := L;
WHEN RSYN THEN Q9 := L;
```

```
test_vectors IN U49
([CLK,CLK1,En_Rmac,Clr_Rmac,Lo_Rmac,DIN,STAT]-> [COUNT,MCNT,Ad6,Ad5,CAR])

[C,C,1,0,1,^h3,0] -> [^h0,^h0,0,0,0];CLEAR
[0,0,0,1,1,^h3,0] -> [^h0,^h0,0,0,0];
[C,C,0,1,1,^h3,0] -> [^h0,^h0,0,0,0];
[C,C,0,1,0,^h3,0] -> [^hC,^h0,0,0,0];LOAD 3
[C,C,0,1,0,^h3,1] -> [^hC,^h0,0,0,0];status
[C,C,1,0,1,^h3,X] -> [^h0,^h0,0,0,0];CLEAR

@radix 16;

@const n=1;
@repeat ^h1f
        {[C,C,1,1,1,^h3,0] -> [^h0+n,^h0,0,0,0];increment
         [C,C,0,1,1,^h3,0] -> [^h0+n,^h0,0,0,0];hold
         [C,C,0,1,1,^h3,0] -> [^h0+n,^h0,0,0,0];hold
                @const n = n+1;}
         [C,C,1,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];(AD5 gen)100
         [C,C,0,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];hold
         [C,C,0,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];hold

@const n=1;
@repeat ^h1E
        {[C,C,1,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];inc(Ad5 gen)
         [C,C,0,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];hold
         [C,C,0,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];hold
                @const n = n+1;}
         [C,C,1,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];inc 193
         [C,C,0,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];hold
         [C,C,0,1,1,^h3,0] -> [^h0+n,^h1,0,1,0];hold
         [C,C,1,1,1,^h3,0] -> [^h0,^h2,1,0,0];inc(Ad6 gen)
         [C,C,0,1,1,^h3,0] -> [^h0,^h2,1,0,0];hold
         [C,C,0,1,1,^h3,0] -> [^h0,^h2,1,0,0];hold

[C,C,0,1,0,^h3,0] -> [^hC,^h2,1,0,0];LOAD 3
[C,C,0,1,0,^h3,1] -> [^hC,^h2,0,0,0];status
@const n=1;
@repeat ^h3
        {[C,C,1,1,1,^h3,1] -> [^hC+n,^h2,0,0,0];increment
         [C,C,0,1,1,^h3,1] -> [^hC+n,^h2,0,0,0];hold
         [C,C,0,1,1,^h3,1] -> [^hC+n,^h2,0,0,0];hold
                @const n = n+1;}
[C,C,0,1,0,^h0,0] -> [^h0,^h2,1,0,0];LOAD 0
[C,C,0,1,0,^h0,0] -> [^h0,^h2,1,0,0];status COMPLETE

@radix 0A;
```

```
test_vectors IN U50
([CLK,CLK1,En_Rmac,SRESET,NEXT,CAR,STAT]-> [MSBCT,MAD,RSYN])

[C,C,1,0,0,0,0] -> [^h1,^h1,1];CLEAR
[C,C,1,0,0,0,0] -> [^h1,^h1,1];CLEAR
[C,C,1,0,0,0,0] -> [^h1,^h1,1];CLEAR
[C,C,0,1,0,0,0] -> [^h1,^h1,0];
[C,C,1,1,0,0,0] -> [^h1,^h1,0];En_Rmac
[C,C,1,1,0,1,0] -> [^h2,^h2,0];En_Rmac-CAR
[C,C,0,1,0,1,0] -> [^h2,^h2,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h2,^h2,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h2,^h2,0];En_Rmac
[C,C,1,1,0,1,0] -> [^h3,^h3,0];En_Rmac-CAR
[C,C,0,1,0,1,0] -> [^h3,^h3,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h3,^h3,0];En_Rmac
[C,C,0,1,0,1,0] -> [^h3,^h3,0];En_Rmac
[C,C,0,1,0,1,1] -> [^h3,^h0,0];En_Rmac-STATUS
[C,C,0,1,0,1,1] -> [^h3,^h0,0];En_Rmac
[C,C,1,1,0,1,0] -> [^h4,^h4,0];En_Rmac
[C,C,1,1,0,1,0] -> [^h5,^h5,0];En_Rmac-CAR
[C,C,0,1,0,1,0] -> [^h5,^h5,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h5,^h5,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h5,^h5,0];En_Rmac
[C,C,1,1,0,1,0] -> [^h6,^h6,0];En_Rmac-CAR
[C,C,0,1,0,1,0] -> [^h6,^h6,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h6,^h6,0];En_Rmac
[C,C,0,1,0,1,0] -> [^h6,^h6,0];En_Rmac
[C,C,0,1,0,1,1] -> [^h6,^h0,0];En_Rmac-STATUS
[C,C,0,1,0,1,1] -> [^h6,^h0,0];En_Rmac
[C,C,1,1,0,1,0] -> [^h7,^h7,0];En_Rmac-CAR
[C,C,0,1,0,1,0] -> [^h7,^h7,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h7,^h7,0];En_Rmac
[0,0,1,1,0,1,0] -> [^h7,^h7,0];En_Rmac
[C,C,1,1,1,1,0] -> [^h1,^h1,0];SKIP 0
[C,C,1,1,0,0,0] -> [^h1,^h1,0];En_Rmac
[C,C,1,1,0,1,0] -> [^h2,^h2,0];En_Rmac-CAR
[C,C,0,1,0,1,0] -> [^h2,^h2,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h2,^h2,0];En_Rmac
[C,C,0,1,0,0,0] -> [^h2,^h2,0];En_Rmac
[C,C,0,0,0,0,0] -> [^h1,^h1,1];CLEAR

test_vectors IN U50
([CLK,CLK1,En_Rmac,SRESET,NEXT,CAR,STAT]-> [MSBCT,MAD,RSYN])

[C,C,1,0,0,0,0] -> [^h1,^h1,1];CLEAR
[C,C,1,1,0,0,0] -> [^h1,^h1,0];
[C,C,1,1,1,1,0] -> [^h2,^h2,0];NEXT & CAR
[C,C,1,1,0,1,0] -> [^h3,^h3,0];EN & CAR
[C,C,0,1,0,0,0] -> [^h3,^h3,0];

end RADCON
```

```
module TXBCT

title RECEIVE MEMORY ADDRESS COUNTER  -- ATCINT U53
Leo J. Wapelhorst---FAA Tech Center-----6-9-90
   U53,U54 device P22V10;

This module contains the Transmit Memory address counter.
It is a 9 bit counter which is preset with the # of words
to be transmitted. It produces an output when it reaches
zero.

CLK,En_Ct,Ld_Ct                          pin in U53 1,2,13;
D0,D1,D2,D3,D4                           pin in U53 3,4,5,6,7;
D5,D6,D7,D8                              pin in U53 8,9,10,11;
Q4,Q3,Q2,Q1,Q0                           pin in U53 23,22,21,20,19;
Q8,Q7,Q6,Q5,BOR                          pin in U53 18,17,16,15,14;

CLK,En_Ct,Ld_Ct                          pin in U54 1,2,13;
BOR,D9                                   pin in U54 3,4;
Q9,WC0                                   pin in U54 23,22;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
DIN   = [D8..D0];
COUNT  = [Q8..Q0];

equations in U53

COUNT := (((COUNT-1)&En_Ct)#COUNT&!En_Ct&Ld_Ct);
BOR    = (COUNT==0);

WHEN !Ld_Ct THEN Q0 := D0;
WHEN !Ld_Ct THEN Q1 := D1;
WHEN !Ld_Ct THEN Q2 := D2;
WHEN !Ld_Ct THEN Q3 := D3;
WHEN !Ld_Ct THEN Q4 := D4;
WHEN !Ld_Ct THEN Q5 := D5;
WHEN !Ld_Ct THEN Q6 := D6;
WHEN !Ld_Ct THEN Q7 := D7;
WHEN !Ld_Ct THEN Q8 := D8;

equations in U54

Q9    := BOR & !En_Ct & Q9 # Q9 & !BOR;;
!WC0  := BOR & !Q9;

WHEN !Ld_Ct THEN Q9 := D9;
```

test_vectors in U53

([CLK,En_Ct,Ld_Ct,DIN]-> [COUNT,BOR])

[C,0,1,^h07] -> [^h00,1];
[C,0,0,^h07] -> [^h07,0];LOAD
[C,0,1,^h07] -> [^h07,0];
[C,0,1,^h07] -> [^h07,0];
[C,1,1,^h07] -> [^h06,0];
[C,0,1,^h07] -> [^h06,0];
[C,0,1,^h07] -> [^h06,0];
[C,1,1,^h07] -> [^h05,0];
[C,0,1,^h07] -> [^h05,0];
[C,1,1,^h07] -> [^h04,0];
[C,0,1,^h07] -> [^h04,0];
[C,0,1,^h07] -> [^h04,0];
[C,1,1,^h07] -> [^h03,0];
[C,0,1,^h07] -> [^h03,0];
[C,0,1,^h07] -> [^h03,0];
[C,1,1,^h07] -> [^h02,0];
[C,0,1,^h07] -> [^h02,0];
[C,0,1,^h07] -> [^h02,0];
[C,1,1,^h07] -> [^h01,0];
[C,0,1,^h07] -> [^h01,0];
[C,0,1,^h07] -> [^h01,0];
[C,1,1,^h07] -> [^h00,1];
[C,0,1,^h07] -> [^h00,1];
[C,0,1,^h07] -> [^h00,1];
[C,0,1,^h07] -> [^h00,1];

test_vectors in U54

([CLK,En_Ct,Ld_Ct,D9,BOR]-> [Q9,WC0])

[C,0,1,1,0] -> [0,1];
[C,0,0,1,0] -> [1,1];LOAD
[C,1,1,1,0] -> [1,1];
[C,1,1,1,0] -> [1,1];
[C,1,1,1,1] -> [0,1];
[C,1,1,1,0] -> [0,1];
[C,1,1,1,0] -> [0,1];
[C,1,1,0,1] -> [0,0];
[C,0,1,0,0] -> [0,1];
[C,0,0,0,0] -> [0,1];LOAD
[C,1,1,0,0] -> [0,1];
[C,0,1,0,1] -> [0,0];
[C,1,1,0,1] -> [0,0];
[C,0,1,0 1] -> [0,0];

end TXBCT

```
module TXCONT

title TRANSMITTER CONTROL LOGIC ATCINT-2  U55,U56
        Leo J. Wapelhorst---FAA Tech Center-----7-17-90

U55,U56 device P22V10;

CLK,TXCKEN,TXACTIVE,MAYBE                pin in U56 1,2,3,4;
DISTXPR                                  pin in U56 13;
ABORT,WC0,SRESET                         pin in U56 5,7,11;
QA,QB,QC,QD,QE                           pin in U56 19,20,21,22,23;
TXINT,LOADSR,TX0S,TX10S                  pin in U56 18,14,16,15;
L                                        pin in U56 17;to delay loads

reset,preset                             node in U56 25,26;

CLK,TXCKEN,TXSTART,WC0                   pin in U55 1,2,3,4;
QA,QB,QC,QD,QE                           pin in U55 5,6,7,8,9;
SRESET                                   pin in U55 11;
TXACTIVE,MAYBE,En_Memad                  pin in U55 23,22,17;
ENTXMEM,En_Tct,Ld_Tct                    pin in U55 16,15,14;
TXST1,TXST2                              pin in U55 18,19;


reset,preset            node in U55 25,26;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
ST = [QE,QD,QC,QB,QA];

 Explicitly state macro cell configuration

TXINT Istype in U56 neg;
LOADSR Istype in U56 pos;
TX0S Istype in U56 neg;

Counter states

S0 = ^b00000;   S1 = ^b00001;   S2 = ^b00011;   S3 = ^b00010;
S4 = ^b00110;   S5 = ^b00111;   S6 = ^b00101;   S7 = ^b00100;
S8 = ^b01100;   S9 = ^b01101;   S10 = ^b01111;  S11 = ^b01110;
S12 = ^b01010;  S13 = ^b01011;  S14 = ^b01001;  S15 = ^b01000;
S16 = ^b10000;  S17 = ^b10001;  S18 = ^b10011;  S19 = ^b10010;
S20 = ^b10110;  S21 = ^b10111;  S22 = ^b10101;  S23 = ^b10100;
S24 = ^b11100;  S25 = ^b11101;  S26 = ^b11111;  S27 = ^b11110;
```

equations in U56

```
preset        = !TXACTIVE;

LOADSR  = L;
L     := (WC0 &
        ((TXCKEN & ((ST==^h13) # (ST==^h1E))) & !DISTXPR #
         (TXCKEN & ((ST==^h13) # (ST==^h1F))) & DISTXPR  #
         LOADSR & !TXCKEN));
TXINT := !WC0 & TXCKEN & (ST==^h12);
TX0S := TXCKEN & (ST==0) # hex 0
     !TXCKEN & TX0S    #
      TXCKEN & TX0S & (ST!=^h13);
TX10S := !TXACTIVE & (ST==0) #
      TX10S & !TXCKEN     #
      TXCKEN & (ST==^h11)  #
      TXACTIVE & !WC0 & (ST==^h12);;
```

equations in U55

```
TXACTIVE := SRESET & (!TXST1 & TXST2 # TXACTIVE & !TXCKEN #
         TXACTIVE & TXCKEN & WC0 #
         TXACTIVE & (ST!=^h12));
!ENTXMEM  = TXACTIVE & ((ST==^h13) # (ST==^h1E) #
                (ST==^h12) # (ST==^h16));
!ENTXMEM   = TXACTIVE;
En_Memad  = TXCKEN & (ST==^h16);
!Ld_Tct   = TXCKEN & (ST==^h3);
En_Tct    = TXCKEN &  (ST==^h17);
TXST1   := TXSTART;
TXST2   := TXST1;
```

state_diagram in U56 [QE,QD,QC,QB,QA]

```
State S0:       case TXCKEN: S1;
                  !TXCKEN: S0;
 endcase;
State S1:       case TXCKEN: S2;
                  !TXCKEN: S1;
 endcase;
State S2:       case TXCKEN: S3;
                  !TXCKEN: S2;
 endcase;
State S3:       case TXCKEN: S4;
                  !TXCKEN: S3;
 endcase;
State S4:       case TXCKEN: S5;
                  !TXCKEN: S4;
 endcase;
State S5:       case TXCKEN: S6;
                  !TXCKEN: S5;
 endcase;
State S6:       case TXCKEN: S7;
                  !TXCKEN: S6;
 endcase;
State S7:       case TXCKEN: S8;
                  !TXCKEN: S7;
 endcase;
State S8:       case TXCKEN: S9;
                  !TXCKEN: S8;
 endcase;
State S9:       case TXCKEN: S10;
                  !TXCKEN: S9;
 endcase;
State S10:case TXCKEN: S11;
                  !TXCKEN: S10;
 endcase;
State S11:case TXCKEN: S12;
                  !TXCKEN: S11;
 endcase;
State S12:case TXCKEN: S13;
                  !TXCKEN: S12;
 endcase;
State S13:case TXCKEN: S14;
                  !TXCKEN: S13;
 endcase;
State S14:case TXCKEN: S15;
                  !TXCKEN: S14;
 endcase;
State S15:case TXCKEN: S16;
                  !TXCKEN: S15;
 endcase;
State S16:case TXCKEN: S17;
                  !TXCKEN: S16;
 endcase;
```

```
State S17:case TXCKEN: S18;
                       !TXCKEN: S17;
 endcase;
State S18:case TXCKEN: S19;
                       !TXCKEN: S18;
 endcase;
State S19:case TXCKEN: S20;
                       !TXCKEN: S19;
 endcase;
State S20:case TXCKEN: S21;
                       !TXCKEN. S20;
 endcase;
State S21:case TXCKEN: S22;
                       !TXCKEN: S21;
 endcase;
State S22:case TXCKEN: S23;
                       !TXCKEN: S22;
 endcase;
State S23:case TXCKEN: S24;
                       !TXCKEN: S23;
 endcase;
State S24:case TXCKEN: S25;
                       !TXCKEN: S24;
 endcase;
State S25:case TXCKEN: S26;
                       !TXCKEN: S25;
 endcase;
State S26:case TXCKEN & !DISTXPR: S27;
                             TXCKEN & DISTXPR: S19;
                       !TXCKEN: S26;
 endcase;
State S27:case TXCKEN: S19;
                       !TXCKEN: S27;
 endcase;
```

test_vectors in U56

([TXCKEN,CLK,SRESET,TXACTIVE,WC0,DISTXPR]-> [ST,TXINT, LOADSR,TX0S,TX10S])

[1,C,1,0,1,0] -> [^h00,0,1,0,1];
[1,C,1,0,1,0] -> [^h00,0,1,0,1];
[0,C,1,0,1,0] -> [^h00,0,1,0,1];
[1,C,1,1,1,0] -> [^h01,0,0,1,0];TXACTIVE,tx0s->1,tx10->0
[0,C,1,1,1,0] -> [^h01,0,0,1,0];
[0,C,1,1,1,0] -> [^h01,0,0,1,0];
[1,C,1,1,1,0] -> [^h03,0,0,1,0];transmitting zeros
[0,C,1,1,1,0] -> [^h03,0,0,1,0];
[1,C,1,1,1,0] -> [^h02,0,0,1,0];
[0,C,1,1,1,0] -> [^h02,0,0,1,0];
[1,C,1,1,1,0] -> [^h06,0,0,1,0];
[0,C,1,1,1,0] -> [^h06,0,0,1,0];
[0,C,1,1,1,0] -> [^h06,0,0,1,0];
[1,C,1,1,1,0] -> [^h07,0,0,1,0];
[0,C,1,1,1,0] -> [^h07,0,0,1,0];
[0,C,1,1,1,0] -> [^h07,0,0,1,0];
[1,C,1,1,1,0] -> [^h05,0,0,1,0];
[0,C,1,1,1,0] -> [^h05,0,0,1,0];
[1,C,1,1,1,0] -> [^h04,0,0,1,0];
[0,C,1,1,1,0] -> [^h04,0,0,1,0];V20
[0,C,1,1,1,0] -> [^h04,0,0,1,0];
[1,C,1,1,1,0] -> [^h0C,0,0,1,0];
[0,C,1,1,1,0] -> [^h0c,0,0,1,0];
[0,C,1,1,1,0] -> [^h0c,0,0,1,0];
[1,C,1,1,1,0] -> [^h0d,0,0,1,0];
[0,C,1,1,1,0] -> [^h0d,0,0,1,0];
[1,C,1,1,1,0] -> [^h0f,0,0,1,0];
[0,C,1,1,1,0] -> [^h0f,0,0,1,0];
[1,C,1,1,1,0] -> [^h0e,0,0,1,0];
[0,C,1,1,1,0] -> [^h0e,0,0,1,0];
[0,C,1,1,1,0] -> [^h0c,0,0,1,0];
[1,C,1,1,1,0] -> [^h0a,0,0,1,0];
[0,C,1,1,1,0] -> [^h0a,0,0,1,0];
[1,C,1,1,1,0] -> [^h0b,0,0,1,0];
[0,C,1,1,1,0] -> [^h0b,0,0,1,0];
[1,C,1,1,1,0] -> [^h09,0,0,1,0];
[0,C,1,1,1,0] -> [^h09,0,0,1,0];
[1,C,1,1,1,0] -> [^h08,0,0,1,0];
[0,C,1,1,1,0] -> [^h08,0,0,1,0];
[1,C,1,1,1,0] -> [^h10,0,0,1,0];V40
[0,C,1,1,1,0] -> [^h10,0,0,1,0];
[1,C,1,1,1,0] -> [^h11,0,0,1,0];
[0,C,1,1,1,0] -> [^h11,0,0,1,0];
[0,C,1,1,1,0] -> [^h11,0,0,1,0];
[0,C,1,1,1,0] -> [^h11,0,0,1,0];
[1,C,1,1,1,0] -> [^h13,0,0,1,1];transmit a 1
[0,C,1,1,1,0] -> [^h13,0,0,1,1];
[0,C,1,1,1,0] -> [^h13,0,0,1,1];
[0,C,1,1,1,0] -> [^h13,0,0,1,1];
[1,C,1,1,1,0] -> [^h12,0,1,0,0];load sr

```
[0,C,1,1,1,0] ->  [^h12,0,1,0,0];
[1,C,1,1,1,0] ->  [^h16,0,0,0,0];
[0,C,1,1,1,0] ->  [^h16,0,0,0,0];
[0,C,1,1,1,0] ->  [^h16,0,0,0,0];
[1,C,1,1,1,0] ->  [^h17,0,0,0,0];
[0,C,1,1,1,0] ->  [^h17,0,0,0,0];
[0,C,1,1,1,0] ->  [^h17,0,0,0,0];
[1,C,1,1,1,0] ->  [^h15,0,0,0,0];
[0,C,1,1,1,0] ->  [^h15,0,0,0,0];
[1,C,1,1,1,0] ->  [^h14,0,0,0,0];
[0,C,1,1,1,0] ->  [^h14,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1C,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1c,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1d,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1d,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1f,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1f,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1e,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1e,0,0,0,0];
[1,C,1,1,1,0] ->  [^h12,0,1,0,0];load sr
[0,C,1,1,1,0] ->  [^h12,0,1,0,0];
[0,C,1,1,1,0] ->  [^h12,0,1,0,0];
[1,C,1,1,1,0] ->  [^h16,0,0,0,0];
[0,C,1,1,1,0] ->  [^h16,0,0,0,0];
[1,C,1,1,1,0] ->  [^h17,0,0,0,0];
[0,C,1,1,1,0] ->  [^h17,0,0,0,0];
[1,C,1,1,1,0] ->  [^h15,0,0,0,0];
[0,C,1,1,1,0] ->  [^h15,0,0,0,0];
[1,C,1,1,1,0] ->  [^h14,0,0,0,0];
[0,C,1,1,1,0] ->  [^h14,0,0,0,0];V80
[1,C,1,1,1,0] ->  [^h1c,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1c,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1d,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1d,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1d,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1f,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1f,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1e,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1e,0,0,0,0];
[1,C,1,1,1,0] ->  [^h12,0,1,0,0];load sr
[0,C,1,1,1,0] ->  [^h12,0,1,0,0];
[1,C,1,1,1,0] ->  [^h16,0,0,0,0];
[0,C,1,1,1,0] ->  [^h16,0,0,0,0];
[1,C,1,1,1,0] ->  [^h17,0,0,0,0];
[0,C,1,1,1,0] ->  [^h17,0,0,0,0];
[0,C,1,1,1,0] ->  [^h17,0,0,0,0];
[0,C,1,1,1,0] ->  [^h17,0,0,0,0];
[1,C,1,1,1,0] ->  [^h15,0,0,0,0];
[0,C,1,1,1,0] ->  [^h15,0,0,0,0];
[1,C,1,1,1,0] ->  [^h14,0,0,0,0];V100
[0,C,1,1,1,0] ->  [^h14,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1C,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1c,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1d,0,0,0,0];
```

```
[0,C,1,1,1,0] ->  [^h1d,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1f,0,0,0,0],
[0,C,1.1,1,0] ->  [^h1f,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1e,0,0,0,0];
[0,C,1,1.1,0] ->  [^h1e,0,0,0,0];
[1,C,1,1,1,0] ->  [^h12,0,1,0,0];load sr, EOM decoded
[0,C,1,1,1,0] ->  [^h12,0,1,0,0];
[1,C,1,1,1,0] ->  [^h1o,0,0,0,0];
[0,C,1,1,1,0] ->  [^h16,0,0,0,0];
[0,C,1,1,1,0] ->  [^h16,0,0,0,0];
[1,C,1,1,1,0] ->  [^h17,0,0,0,0];
[0,C,1,1,1,0] ->  [^h17,0,0,0,0];
[1,C,1,1,1,0] ->  [^h15,0,0,0,0];
[0,C,1,1.1,0] ->  [^h15,0,0,0,0];
[1,C,1,1,1,0] ->  [^h14,0,0,0,0];
[0,C,1,1,1,0] ->  [^h14,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1c,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1c,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1d,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1d,0,0,0,0];
[1,C,1,1,1,0] ->  [^h1f,0,0,0,0];
[0,C,1,1,1,0] ->  [^h1f,0,0,0,0];
[1,C,1,1.0,0] ->  [^h1e,0,0,0,0];
[0,C,1,1,0,0] ->  [^h1e.0,0,0,0];
[1,C,1,1,0,0] ->  [^h12,0,0,0,0];TXINT GENERATED
[0,C,1,1,0,0] ->  [^h12,0,0,0,1]:V130
[0,C,1,1,0,0] ->  [^h12,0,0,0,1];
[1,C,1,1,0,0] ->  [^h16,1,0,0,1];
[1,C,1,0,1,0] ->  [^h00,0,1,0,1];
[0,C,1,0,1,0] ->  [^h00,0,1,0,1];
[0,C,1,0,1,0] ->  [^h00,0,1,0,1];
```

test_vectors in U56 TRANSMIT PROCESSING DISABLED

([TXCKEN,CLK,SRESET,TXACTIVE,WC0,DISTXPR]-> [ST,TXINT, LOADSR,TX0S,TX10S])

[1,C,1,0,1,1] -> [^h00,0,1,0,1];
[1,C,1,0,1,1] -> [^h00,0,1,0,1];
[0,C,1,0,1,1] -> [^h00,0,1,0,1];
[1,C,1,1,1,1] -> [^h01,0,0,1,0];TXACTIVE,tx0s->1,tx10->0
[0,C,1,1,1,1] -> [^h01,0,0,1,0];
[0,C,1,1,1,1] -> [^h01,0,0,1,0];
[1,C,1,1,1,1] -> [^h03,0,0,1,0];transmitting zeros
[0,C,1,1,1,1] -> [^h03,0,0,1,0];
[1,C,1,1,1,1] -> [^h02,0,0,1,0];
[0,C,1,1,1,1] -> [^h02,0,0,1,0];
[1,C,1,1,1,1] -> [^h06,0,0,1,0];
[0,C,1,1,1,1] -> [^h06,0,0,1,0];
[0,C,1,1,1,1] -> [^h06,0,0,1,0];
[1,C,1,1,1,1] -> [^h07,0,0,1,0];
[0,C,1,1,1,1] -> [^h07,0,0,1,0];
[0,C,1,1,1,1] -> [^h07,0,0,1,0];
[1,C,1,1,1,1] -> [^h05,0,0,1,0];
[0,C,1,1,1,1] -> [^h05,0,0,1,0];
[1,C,1,1,1,1] -> [^h04,0,0,1,0];
[0,C,1,1,1,1] -> [^h04,0,0,1,0];
[0,C,1,1,1,1] -> [^h04,0,0,1,0];
[1,C,1,1,1,1] -> [^h0c,0,0,1,0];
[0,C,1,1,1,1] -> [^h0c,0,0,1,0];
[0,C,1,1,1,1] -> [^h0c,0,0,1,0];
[1,C,1,1,1,1] -> [^h0d,0,0,1,0];
[0,C,1,1,1,1] -> [^h0d,0,0,1,0];
[1,C,1,1,1,1] -> [^h0f,0,0,1,0];
[0,C,1,1,1,1] -> [^h0f,0,0,1,0];
[1,C,1,1,1,1] -> [^h0e,0,0,1,0];
[0,C,1,1,1,1] -> [^h0e,0,0,1,0];
[0,C,1,1,1,1] -> [^h0e,0,0,1,0];
[1,C,1,1,1,1] -> [^h0a,0,0,1,0];
[0,C,1,1,1,1] -> [^h0a,0,0,1,0];
[1,C,1,1,1,1] -> [^h0b,0,0,1,0];
[0,C,1,1,1,1] -> [^h0b,0,0,1,0];
[1,C,1,1,1,1] -> [^h09,0,0,1,0];
[0,C,1,1,1,1] -> [^h09,0,0,1,0];
[1,C,1,1,1,1] -> [^h08,0,0,1,0];
[0,C,1,1,1,1] -> [^h08,0,0,1,0];
[1,C,1,1,1,1] -> [^h10,0,0,1,0];
[0,C,1,1,1,1] -> [^h10,0,0,1,0];
[1,C,1,1,1,1] -> [^h11,0,0,1,0];
[0,C,1,1,1,1] -> [^h11,0,0,1,0];
[0,C,1,1,1,1] -> [^h11,0,0,1,0];
[0,C,1,1,1,1] -> [^h11,0,0,1,0];
[1,C,1,1,1,1] -> [^h13,0,0,1,1];transmit a 1 TIME
[0,C,1,1,1,1] -> [^h13,0,0,1,1];
[0,C,1,1,1,1] -> [^h13,0,0,1,1];
[0,C,1,1,1,1] -> [^h13,0,0,1,1];
[1,C,1,1,1,1] -> [^h12,0,1,0,0];load sr

[0,C,1,1,1,1] -> [^h12,0,1,0,0];
[1,C,1,1,1,1] -> [^h16,0,0,0,0];
[0,C,1,1,1,1] -> [^h16,0,0,0,0];
[0,C,1,1,1,1] -> [^h16,0,0,0,0];
[1,C,1,1,1,1] -> [^h17,0,0,0,0];
[0,C,1,1,1,1] -> [^h17,0,0,0,0];
[0,C,1,1,1,1] -> [^h17,0,0,0,0];
[1,C,1,1,1,1] -> [^h15,0,0,0,0];
[0,C,1,1,1,1] -> [^h15,0,0,0,0];
[1,C,1,1,1,1] -> [^h14,0,0,0,0];
[0,C,1,1,1,1] -> [^h14,0,0,0,0];
[1,C,1,1,1,1] -> [^h1c,0,0,0,0];
[0,C,1,1,1,1] -> [^h1c,0,0,0,0];
[1,C,1,1,1,1] -> [^h1d,0,0,0,0];
[0,C,1,1,1,1] -> [^h1d,0,0,0,0];
[1,C,1,1,1,1] -> [^h1f,0,0,0,0];
[0,C,1,1,1,1] -> [^h1f,0,0,0,0];
[1,C,1,1,1,1] -> [^h12,0,1,0,0];LOADSR
[0,C,1,1,1,1] -> [^h12,0,1,0,0];
[0,C,1,1,1,1] -> [^h12,0,1,0,0];
[1,C,1,1,1,1] -> [^h16,0,0,0,0];
[0,C,1,1,1,1] -> [^h16,0,0,0,0];
[1,C,1,1,1,1] -> [^h17,0,0,0,0];
[0,C,1,1,1,1] -> [^h17,0,0,0,0];
[1,C,1,1,1,1] -> [^h15,0,0,0,0];
[0,C,1,1,1,1] -> [^h15,0,0,0,0];
[1,C,1,1,1,1] -> [^h14,0,0,0,0];
[0,C,1,1,1,1] -> [^h14,0,0,0,0];
[1,C,1,1,1,1] -> [^h1C,0,0,0,0];
[0,C,1,1,1,1] -> [^h1c,0,0,0,0];
[1,C,1,1,1,1] -> [^h1d,0,0,0,0];
[0,C,1,1,1,1] -> [^h1d,0,0,0,0];
[0,C,1,1,1,1] -> [^h1d,0,0,0,0];
[1,C,1,1,1,1] -> [^h1f,0,0,0,0];
[0,C,1,1,1,1] -> [^h1f,0,0,0,0];
[1,C,1,1,1,1] -> [^h12,0,1,0,0];load sr
[0,C,1,1,1,1] -> [^h12,0,1,0,0];
[1,C,1,1,1,1] -> [^h16,0,0,0,0];
[0,C,1,1,1,1] -> [^h16,0,0,0,0];

test_vectors in U55

([TXCKEN,CLK,SRESET,ST,TXSTART,WC0]-> [TXACTIVE,En_Memad,En_Tct,Ld_Tct])

[1,C,1,^h00,1,1] -> [0,0,0,1];
[1,C,1,^h00,1,1] -> [0,0,0,1];
[0,C,1,^h00,0,1] -> [0,0,0,1];TXSTART OCCURS-TXACTIVE
[0,C,1,^h00,0,1] -> [1,0,0,1];
[0,C,1,^h00,0,1] -> [1,0,0,1];
[1,C,1,^h01,0,1] -> [1,0,0,1];tx0s->1,tx10->0
[0,C,1,^h01,0,1] -> [1,0,0,1];
[0,C,1,^h01,0,1] -> [1,0,0,1];
[1,C,1,^h03,0,1] -> [1,0,0,0];transmitting zeros
[0,C,1,^h03,0,1] -> [1,0,0,1];

```
[1,C,1,^h02,0,1] ->     [1,0,0,1];
[0,C,1,^h02,0,1] ->     [1,0,0,1];
[1,C,1,^h06,0,1] ->     [1,0,0,1];En_Memad
[0,C,1,^h06,0,1] ->     [1,0,0,1];
[0,C,1,^h06,0,1] ->     [1,0,0,1];
[1,C,1,^h07,0,1] ->     [1,0,0,1];
[0,C,1,^h07,0,1] ->     [1,0,0,1];
[0,C,1,^h07,1,1] ->     [1,0,0,1];
[1,C,1,^h05,1,1] ->     [1,0,0,1];
[0,C,1,^h05,1,1] ->     [1,0,0,1];
[1,C,1,^h04,1,1] ->     [1,0,0,1];
[0,C,1,^h04,1,1] ->     [1,0,0,1];V20
[0,C,1,^h04,1,1] ->     [1,0,0,1];
[1,C,1,^h0c,1,1] ->     [1,0,0,1];
[0,C,1,^h0c,1,1] ->     [1,0,0,1];
[0,C,1,^h0c,1,1] ->     [1,0,0,1];
[1,C,1,^h0d,1,1] ->     [1,0,0,1];
[0,C,1,^h0d,1,1] ->     [1,0,0,1];
[1,C,1,^h0f,1,1] ->     [1,0,0,1];
[0,C,1,^h0f,1,1] ->     [1,0,0,1];
[1,C,1,^h0e,1,1] ->     [1,0,0,1];
[0,C,1,^h0e,1,1] ->     [1,0,0,1];
[0,C,1,^h0e,1,1] ->     [1,0,0,1];
[1,C,1,^h0a,1,1] ->     [1,0,0,1];
[0,C,1,^h0a,1,1] ->     [1,0,0,1];
[1,C,1,^h0b,1,1] ->     [1,0,0,1];
[0,C,1,^h0b,1,1] ->     [1,0,0,1];
[1,C,1,^h09,1,1] ->     [1,0,0,1];
[0,C,1,^h09,1,1] ->     [1,0,0,1];
[1,C,1,^h08,1,1] ->     [1,0,0,1];
[0,C,1,^h08,1,1] ->     [1,0,0,1];
[1,C,1,^h10,1,1] ->     [1,0,0,1];V40
[0,C,1,^h10,1,1] ->     [1,0,0,1];
[1,C,1,^h11,1,1] ->     [1,0,0,1];
[0,C,1,^h11,1,1] ->     [1,0,0,1];
[0,C,1,^h11,1,1] ->     [1,0,0,1];
[0,C,1,^h11,1,1] ->     [1,0,0,1];
[1,C,1,^h13,1,1] ->     [1,0,0,1];transmit a 1
[0,C,1,^h13,1,1] ->     [1,0,0,1];
[0,C,1,^h13,1,1] ->     [1,0,0,1];
[0,C,1,^h13,1,1] ->     [1,0,0,1];
[1,C,1,^h12,1,1] ->     [1,0,0,1];load sr
[0,C,1,^h12,1,1] ->     [1,0,0,1];
[1,C,1,^h16,1,1] ->     [1,1,0,1];
[0,C,1,^h16,1,1] ->     [1,0,0,1];En_Memad
[0,C,1,^h16,1,1] ->     [1,0,0,1];
[1,C,1,^h17,1,1] ->     [1,0,1,1];
[0,C,1,^h17,1,1] ->     [1,0,0,1];
[0,C,1,^h17,1,1] ->     [1,0,0,1];
[1,C,1,^h15,1,1] ->     [1,0,0,1];
[0,C,1,^h15,1,1] ->     [1,0,0,1];
[1,C,1,^h14,1,1] ->     [1,0,0,1];
[0,C,1,^h14,1,1] ->     [1,0,0,1];
[1,C,1,^h1c,1,1] ->     [1,0,0,1];
```

[0,C,1,^h1c,1,1] -> [1,0,0,1];
[1,C,1,^h1d,1,1] -> [1,0,0,1];
[0,C,1,^h1d,1,1] -> [1,0,0,1];
[1,C,1,^h1f,1,1] -> [1,0,0,1];
[0,C,1,^h1f,1,1] -> [1,0,0,1];
[1,C,1,^h1e,1,1] -> [1,0,0,1];
[0,C,1,^h1e,1,1] -> [1,0,0,1];
[1,C,1,^h12,1,1] -> [1,0,0,1];load sr
[0,C,1,^h12,1,1] -> [1,0,0,1];
[0,C,1,^h12,1,1] -> [1,0,0,1];
[1,C,1,^h16,1,1] -> [1,1,0,1];En_Memad
[0,C,1,^h16,1,1] -> [1,0,0,1];
[1,C,1,^h17,1,1] -> [1,0,1,1];
[0,C,1,^h17,1,1] -> [1,0,0,1];
[1,C,1,^h15,1,1] -> [1,0,0,1];
[0,C,1,^h15,1,1] -> [1,0,0,1];
[1,C,1,^h14,1,1] -> [1,0,0,1];
[0,C,1,^h14,1,1] -> [1,0,0,1];
[1,C,1,^h1c,1,1] -> [1,0,0,1];
[0,C,1,^h1c,1,1] -> [1,0,0,1];
[1,C,1,^h1d,1,1] -> [1,0,0,1];
[0,C,1,^h1d,1,1] -> [1,0,0,1];
[0,C,1,^h1d,1,1] -> [1,0,0,1];
[1,C,1,^h1f,1,1] -> [1,0,0,1];
[0,C,1,^h1f,1,1] -> [1,0,0,1];
[1,C,1,^h1e,1,1] -> [1,0,0,1];
[0,C,1,^h1e,1,1] -> [1,0,0,1];
[1,C,1,^h12,1,1] -> [1,0,0,1];load sr
[0,C,1,^h12,1,1] -> [1,0,0,1];
[1,C,1,^h16,1,1] -> [1,1,0,1];En_Memad
[0,C,1,^h16,1,1] -> [1,0,0,1];
[1,C,1,^h17,1,1] -> [1,0,1,1];
[0,C,1,^h17,1,1] -> [1,0,0,1];
[0,C,1,^h17,1,1] -> [1,0,0,1];
[0,C,1,^h17,1,1] -> [1,0,0,1];
[1,C,1,^h15,1,1] -> [1,0,0,1];
[0,C,1,^h15,1,1] -> [1,0,0,1];
[1,C,1,^h14,1,1] -> [1,0,0,1];
[0,C,1,^h14,1,1] -> [1,0,0,1];
[1,C,1,^h1c,1,1] -> [1,0,0,1];
[0,C,1,^h1c,1,1] -> [1,0,0,1];
[1,C,1,^h1d,1,1] -> [1,0,0,1];
[0,C,1,^h1d,1,1] -> [1,0,0,1];
[1,C,1,^h1f,1,1] -> [1,0,0,1];
[0,C,1,^h1f,1,1] -> [1,0,0,1];
[1,C,1,^h1e,1,1] -> [1,0,0,1];
[0,C,1,^h1e,1,1] -> [1,0,0,1];
[1,C,1,^h12,1,1] -> [1,0,0,1];load sr, EOM decoded
[0,C,1,^h12,1,1] -> [1,0,0,1];
[1,C,1,^h16,1,1] -> [1,1,0,1];En_Memad
[0,C,1,^h16,1,1] -> [1,0,0,1];
[0,C,1,^h16,1,1] -> [1,0,0,1];
[1,C,1,^h17,1,1] -> [1,0,1,1];
[0,C,1,^h17,1,1] -> [1,0,0,1];

```
[1,C,1,^h15,1,1] ->        [1,0,0,1];
[0,C,1,^h15,1,1] ->        [1,0,0,1];
[1,C,1,^h14,1,1] ->        [1,0,0,1];
[0,C,1,^h14,1,1] ->        [1,0,0,1];
[1,C,1,^h1c,1,1] ->        [1,0,0,1];
[0,C,1,^h1c,1,0] ->        [1,0,0,1];
[1,C,1,^h1c,1,0] ->        [1,0,0,1];
[0,C,1,^h1d,1,0] ->        [1,0,0,1];
[1,C,1,^h1f,1,0] ->        [1,0,0,1];
[0,C,1,^h1f,1,0] ->        [1,0,0,1];
[1,C,1,^h1f,1,0] ->        [1,0,0,1];
[0,C,1,^h1f,1,0] ->        [1,0,0,1];
[1,C,1,^h12,1,0] ->        [0,0,0,1];TXINT GENERATED
[0,C,1,^h12,1,0] ->        [0,0,0,1];
[1,C,1,^h00,1,0] ->        [0,0,0,1];
[0,C,1,^h00,1,0] ->        [0,0,0,1];
[0,C,1,^h00,1,0] ->        [0,0,0,1];

end TXCONT;
```

```
module TXOUTSR

title TRANSMITTER OUTPUT SHIFT REGISTER ATCINT U57
      Leo J. Wapelhorst---FAA Tech Center-----7-18-90

U57 device P22V10;

CLK,TXCKEN,LOADSR                          pin 1,2,11;
TD0,TD1,TD2,TD3,TD4,TD5,TD6,TD7            pin 3,4,5,6,7,8,9,10;
Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7                     pin 15,16,17,18,19,20,21,22;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;

equations

Q7 := Q6 & TXCKEN & !LOADSR #
   TD7 & LOADSR & TXCKEN     #
   Q7 & !TXCKEN;

Q6 := Q5 & TXCKEN & !LOADSR #
   TD6 & LOADSR & TXCKEN     #
   Q6 & !TXCKEN;

Q5 := Q4 & TXCKEN & !LOADSR #
   TD5 & LOADSR & TXCKEN     #
   Q5 & !TXCKEN;

Q4 := Q3 & TXCKEN & !LOADSR #
   TD4 & LOADSR & TXCKEN     #
   Q4 & !TXCKEN;

Q3 := Q2 & TXCKEN & !LOADSR #
   TD3 & LOADSR & TXCKEN     #
   Q3 & !TXCKEN;

Q2 := Q1 & TXCKEN & !LOADSR #
   TD2 &  LOADSR & TXCKEN    #
   Q2 & !TXCKEN;

Q1 := Q0 & TXCKEN & !LOADSR #
   TD1 & LOADSR & TXCKEN     #
   Q1 & !TXCKEN;

Q0 :=  TD0 & LOADSR & TXCKEN    #
   Q0 & !TXCKEN;
```

```
test_vectors

([TXCKEN,CLK,LOADSR,TD7,TD6,TD5,TD4,TD3,TD2,TD1,TD0]-
>[Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0])

[1,C,1,0,1,0,1,0,1,0,1]->   [0,1,0,1,0,1,0,1];LOADSR
[1,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,1,0,1,0];SHIFT
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,1,0,1,0];
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,1,0,1,0];
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,1,0,1,0];
[1,C,0,0,0,0,0,0,0,0,1]->   [0,1,0,1,0,1,0,0];SHIFT
[0,C,0,0,0,0,0,0,0,0,1]->   [0,1,0,1,0,1,0,0];
[0,C,0,0,0,0,0,0,0,0,1]->   [0,1,0,1,0,1,0,0];
[0,C,0,0,0,0,0,0,0,0,1]->   [0,1,0,1,0,1,0,0];
[1,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,1,0,0,0];SHIFT
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,1,0,0,0];
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,1,0,0,0];
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,1,0,0,0];
[1,C,0,0,0,0,0,0,0,0,0]->   [0,1,0,1,0,0,0,0];SHIFT
[0,C,0,0,0,0,0,0,0,0,0]->   [0,1,0,1,0,0,0,0];
[0,C,0,0,0,0,0,0,0,0,0]->   [0,1,0,1,0,0,0,0];
[1,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,0,0,0,0];SHIFT
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,0,0,0,0];
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,1,0,0,0,0,0];
[1,C,0,0,0,0,0,0,0,0,0]->   [0,1,0,0,0,0,0,0];
[0,C,0,0,0,0,0,0,0,0,0]->   [0,1,0,0,0,0,0,0];
[1,C,0,0,0,0,0,0,0,0,0]->   [1,0,0,0,0,0,0,0];
[0,C,0,0,0,0,0,0,0,0,0]->   [1,0,0,0,0,0,0,0];
[1,C,1,0,1,0,1,0,1,0,0]->   [0,1,0,1,0,1,0,0];LOADSR

end TXOUTSR;
```

```
module TXPAR
THIS VERSION HAS THE TRANSMIT DISABLE LOGIC.

title TRANSMITTER  PARITY GENERATOR ATCINT U58
        Leo J. Wapelhorst---FAA Tech Center-----7-19-90

U58 device P22V10;

CLK,TXCKEN,DISTXPR                                  pin 1,2,15;
TD0,TD1,TD2,TD3,TD4,TD5,TD6,TD7                     pin 3,4,5,6,7,8,9,10;
TX10S,TX0S,TQ7,LOADSR                               pin 11,23,13,14;
TXPA,PL0,PME,PHI,TXDATA                             pin 22,18,20,16,17;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
DT = [TD7,TD6,TD5,TD4,TD3,TD2,TD1,TD0];

equations

PL0  = TD0 $ TD1 $ TD2;
PME  = TD3 $ TD4 $ TD5;
PHI  = TD6 $ TD7;
TXPA = !(PL0 $ PHI $ PME);

TXDATA := (!DISTXPR &
      (TQ7 & !TX10S & !TX0S & TXCKEN & !LOADSR   #SHIFT
      TXPA & LOADSR & TXCKEN & !TX10S          #PAR
      TX10S & TX0S & TXCKEN))                  #SYNC
      TXDATA & !TXCKEN                        #HOLD
      !TXDATA & TX10S & !TX0S & TXCKEN         #IDLE
      DISTXPR & TQ7 & !TX10S & !TX0S & TXCKEN   #
      DISTXPR & TQ7 & TX10S & TX0S & TXCKEN   ;
      SHIFT IF DISABLED


test_vectors NORMAL MODE

([CLK,DT,TX0S,TX10S,TXCKEN,TQ7,LOADSR,DISTXPR] -> [PHI,PME,PL0,TXPA,TXDATA]);

[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,0];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,0];IDLE
[C,^h00,0,1,1,0,0,0] ->    [0,0,0,1,1];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,1];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,1];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,1];IDLE
[C,^h00,0,1,1,0,0,0] ->    [0,0,0,1,0];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,0];IDLE
[C,^h00,0.1,0,0,0,0] ->    [0,0,0,1,0];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,0];IDLE
[C,^h00,0,1,1,0,0,0] ->    [0,0,0,1,1];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,1];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,1];IDLE
[C,^h00,0,1,1,0,0,0] ->    [0,0,0,1,0];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,0];IDLE
[C,^h00,0,1,0,0,0,0] ->    [0,0,0,1,0];IDLE
```

```
[C,^h00,1,0,1,0,0,0] ->    [0,0,0,1,0];TX0S
[C,^h00,1,0,0,0,0,0] ->    [0,0,0,1,0];TX0S
[C,^h00,1,0,1,0,0,0] ->    [0,0,0,1,0];TX0S
[C,^h00,1,0,0,0,0,0] ->    [0,0,0,1,0];TX0S-V20
[C,^h00,1,0,1,0,0,0] ->    [0,0,0,1,0];TX0S
[C,^h00,1,0,0,0,0,0] ->    [0,0,0,1,0];TX0S
[C,^h00,1,0,1,0,0,0] ->    [0,0,0,1,0];TX0S
[C,^h00,1,0,0,0,0,0] ->    [0,0,0,1,0];TX0S
[C,^h00,1,0,0,0,0,0] ->    [0,0,0,1,0];TX0S
[C,^h00,1,1,1,0,0,0] ->    [0,0,0,1,1];TX1S
[C,^h00,1,1,0,0,0,0] ->    [0,0,0,1,1];TX1S
[C,^h00,1,1,0,0,0,0] ->    [0,0,0,1,1];TX1S
[C,^h33,0,0,1,0,0,0] ->    [0,0,0,1,0];      Parity = 1
[C,^h33,0,0,0,0,0,0] ->    [0,0,0,1,0];      Parity = 1
[C,^h77,0,0,1,0,0,0] ->    [1,0,1,1,0];      Parity = 1
[C,^hbd,0,0,0,0,0,0] ->    [1,1,0,1,0];      Parity = 1
[C,^hbd,0,0,1,1,1,0] ->    [1,1,0,1,1];      Parity = 1,LOAD
[C,^hbd,0,0,0,1,0,0] ->    [1,1,0,1,1];      Parity = 1
[C,^hbd,0,0,0,1,0,0] ->    [1,1,0,1,1];      Parity = 1
[C,^h01,0,0,1,0,0,0] ->    [0,0,1,0,0];      Parity = 0
[C,^h01,0,0,0,0,0,0] ->    [0,0,1,0,0];      Parity = 0
[C,^h01,0,0,0,0,0,0] ->    [0,0,1,0,0];      Parity = 0
[C,^h02,0,0,1,1,0,0] ->    [0,0,1,0,1];      Parity = 0
[C,^h02,0,0,0,1,0,0] ->    [0,0,1,0,1];      Parity = 0 V40
[C,^h02,0,0,0,1,0,0] ->    [0,0,1,0,1];      Parity = 0
[C,^h04,0,0,1,1,0,0] ->    [0,0,1,0,1];      Parity = 0
[C,^h04,0,0,0,1,0,0] ->    [0,0,1,0,1];      Parity = 0
[C,^h04,0,0,0,1,0,0] ->    [0,0,1,0,1];      Parity = 0
[C,^h08,0,0,1,1,0,0] ->    [0,1,0,0,1];      Parity = 0
[C,^h08,0,0,0,1,0,0] ->    [0,1,0,0,1];      Parity = 0
[C,^h08,0,0,0,1,0,0] ->    [0,1,0,0,1];      Parity = 0
[C,^h10,0,0,1,1,1,0] ->    [0,1,0,0,0];      Parity = 0
[C,^h10,0,0,0,1,1,0] ->    [0,1,0,0,0];      Parity = 0
[C,^h10,0,0,0,1,1,0] ->    [0,1,0,0,0];      Parity = 0
[C,^h20,0,0,1,0,0,0] ->    [0,1,0,0,0];      Parity = 0
[C,^h20,0,0,0,0,0,0] ->    [0,1,0,0,0];      Parity = 0
[C,^h20,0,0,0,0,0,0] ->    [0,1,0,0,0];      Parity = 0
[C,^h40,0,0,1,0,0,0] ->    [1,0,0,0,0];      Parity = 0
[C,^h40,0,0,0,0,0,0] ->    [1,0,0,0,0];      Parity = 0
[C,^h40,0,0,0,0,0,0] ->    [1,0,0,0,0];      Parity = 0
[C,^h80,0,0,1,1,0,0] ->    [1,0,0,0,1];      Parity = 0
[C,^h80,0,0,0,1,0,0] ->    [1,0,0,0,1];      Parity = 0
[C,^h80,0,0,0,1,0,0] ->    [1,0,0,0,1];      Parity = 0
[C,^hb1,0,0,1,0,1,0] ->    [1,0,1,1,1];      Parity = 1,LOAD
[C,^hb1,0,0,0,0,1,0] ->    [1,0,1,1,1];      Parity = 1
[C,^hb1,0,0,0,0,1,0] ->    [1,0,1,1,1];      Parity = 1 V62
[C,^h80,0,0,1,1,0,0] ->    [1,0,0,0,1];      Parity = 0
[C,^h80,0,0,0,1,0,0] ->    [1,0,0,0,1];      Parity = 0
[C,^h80,0,0,0,1,0,0] ->    [1,0,0,0,1];      Parity = 0
[C,^h80,0,1,0,X,0,0] ->    [1,0,0,0,1];      Parity = 0
[C,^h80,0,1,1,X,0,0] ->    [1,0,0,0,0];      V67
[C,^h80,0,1,1,X,0,0] ->    [1,0,0,0,1];      Parity = 0
[C,^h80,0,1,1,X,0,0] ->    [1,0,0,0,0];      Parity = 0
[C,^h80,0,1,1,X,0,0] ->    [1,0,0,0,1];      Parity = 0
```

test_vectors DISABLE TRANSMIT PROCESSING MODE

([CLK,DT,TX0S,TX10S,TXCKEN,TQ7,LOADSR,DISTXPR] -> [PHI,PME,PL0,TXPA,TXDATA]):

```
[C,^h00,0,1,1,0,0,1] ->    [0,0,0,1,0];V71
[C,^h00,1,0,1,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,0,0,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,0,1,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,0,0,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,0,1,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,0,0,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,0,1,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,0,0,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,0,0,0,0,1] ->    [0,0,0,1,0];V80
[C,^h00,1,1,1,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,1,0,0,0,1] ->    [0,0,0,1,0];
[C,^h00,1,1,1,1,0,1] ->    [0,0,0,1,1];
[C,^h33,0,0,1,0,0,1] ->    [0,0,0,1,0];
[C,^h33,0,0,1,1,0,1] ->    [0,0,0,1,1];
[C,^h77,0,0,1,0,0,1] ->    [1,0,1,1,0];
[C,^h77,0,0,1,1,0,1] ->    [1,0,1,1,1];
[C,^hff,0,0,1,1,0,1] ->    [0,1,1,1,1];
[C,^hff,0,0,0,1,0,1] ->    [0,1,1,1,1];
[C,^hff,0,0,0,1,0,1] ->    [0,1,1,1,1];V90
[C,^h01,0,0,1,0,0,1] ->    [0,0,1,0,0];
[C,^h01,0,0,0,0,0,1] ->    [0,0,1,0,0];
[C,^h01,0,0,0,0,0,1] ->    [0,0,1,0,0];
[C,^h02,0,0,1,1,0,1] ->    [0,0,1,0,1];V95
[C,^h02,0,0,0,0,0,1] ->    [0,0,1,0,1];
[C,^h02,0,0,0,0,0,1] ->    [0,0,1,0,1];
[C,^h04,0,0,1,0,0,1] ->    [0,0,1,0,0];
[C,^h04,0,0,1,1,0,1] ->    [0,0,1,0,1];
[C,^h04,0,0,0,0,0,1] ->    [0,0,1,0,1];
[C,^h08,0,0,1,1,0,1] ->    [0,1,0,0,1];
[C,^h08,0,0,0,1,0,1] ->    [0,1,0,0,1];
[C,^h08,0,0,1,0,0,1] ->    [0,1,0,0,0];V99
[C,^h10,0,0,1,1,1,1] ->    [0,1,0,0,1];
[C,^h10,0,0,0,1,1,1] ->    [0,1,0,0,1];
[C,^h10,0,0,0,1,1,1] ->    [0,1,0,0,1];
[C,^h20,0,0,1,0,0,1] ->    [0,1,0,0,0];
[C,^h20,0,0,0,0,0,1] ->    [0,1,0,0,0];
[C,^h20,0,0,0,0,0,1] ->    [0,1,0,0,0];
[C,^h40,0,0,1,0,0,1] ->    [1,0,0,0,0];
[C,^h40,0,0,0,0,0,1] ->    [1,0,0,0,0];
[C,^h40,0,0,0,0,0,1] ->    [1,0,0,0,0];
[C,^h80,0,0,1,1,0,1] ->    [1,0,0,0,1];
[C,^h80,0,0,0,1,0,1] ->    [1,0,0,0,1];
[C,^h80,0,0,0,1,0,1] ->    [1,0,0,0,1];
[C,^hb1,0,0,1,0,1,1] ->    [1,0,1,1,0];  ,LOAD
[C,^hb1,0,0,0,0,1,1] ->    [1,0,1,1,0];
[C,^hb1,0,0,0,0,1,1] ->    [1,0,1,1,0];
[C,^h80,0,0,1,1,0,1] ->    [1,0,0,0,1];
[C,^h80,0,0,0,1,0,1] ->    [1,0,0,0,1];
```

```
[C,^h80,0,0,0,1,0,1] ->     [1,0,0,0,1];
[C,^h80,0,0,0,1,0,1] ->     [1,0,0,0,1];
[C,^h80,0,0,0,1,0,1] ->     [1,0,0,0,1];

end TXPAR;
module CLKMON

title EXTERNAL CLOCK MONITOR LOGIC  -- ATCINT U60
Leo J. Wapelhorst---FAA Tech Center-----6-22-90
   U60 device P22V10;

Thia module checks for the presence of External Transmit
and receive clocks. This is done in the following manner.
Each occurrence of the internally generated ICLK causes a
1 to be shifted down a shift register. Each external
clock resets the shift register. If the 1 ever reaches
the third stage of the shift register, the output is high
signalling the absence of external clocks. This output is
written into the hardware status latch so that this fact
is accessible to the computer.

CLK,ICLK,TXCKEN,RXCKEN                    pin 1,2,3,4;
NOTXCK,TQ2,TQ1                            pin 23,21,20;
NORXCK,RQ2,RQ1                            pin 22,19,18;
ICLK1,ICLK2                               pin 14,15;
SRESET                                    pin 11;
reset,preset                              node 25,26;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
TMON  = [NOTXCK,TQ2,TQ1];
RMON  = [NORXCK,RQ2,RQ1];
TMON_re = [NOTXCK.re,TQ2.re,TQ1.re];
RMON_re = [NORXCK.re,RQ2.re,RQ1.re];

equations

reset   = !SRESET;
ICLK1  := ICLK;
ICLK2  := ICLK1;
TQ1    := !TXCKEN & (ICLK1 & !ICLK2 # TQ1);
TQ2    := !TXCKEN & (ICLK1 & !ICLK2 & TQ1 # TQ2);
NOTXCK := !TXCKEN & (ICLK1 & !ICLK2 & TQ2 # NOTXCK);

RQ1    := !RXCKEN & (ICLK1 & !ICLK2 # RQ1);
RQ2    := !RXCKEN & (ICLK1 & !ICLK2 & RQ1 # RQ2);
NORXCK := !RXCKEN & (ICLK1 & !ICLK2 & RQ2 # NORXCK);
```

test_vectors

([SRESET,CLK,ICLK,TXCKEN,RXCKEN] -> [TMON,RMON,ICLK1,ICLK2])

[0,C,0,0,0] -> [0,0,0,0];
[1,C,0,0,0] -> [0,0,0,0];
[1,C,0,0,0] -> [0,0,0,0];
[1,C,1,0,0] -> [0,0,1,0];ICLK
[1,C,1,0,0] -> [1,1,1,1];
[1,C,1,0,0] -> [1,1,1,1];
[1,C,1,0,0] -> [1,1,1,1];
[1,C,0,0,0] -> [1,1,0,1];
[1,C,0,0,0] -> [1,1,0,0];
[1,C,0,0,0] -> [1,1,0,0];
[1,C,1,0,0] -> [1,1,1,0];ICLK
[1,C,1,0,0] -> [3,3,1,1];
[1,C,1,0,0] -> [3,3,1,1];
[1,C,1,0,0] -> [3,3,1,1];
[1,C,1,0,0] -> [3,3,1,1];
[1,C,0,0,0] -> [3,3,0,1];
[1,C,0,0,0] -> [3,3,0,0];
[1,C,0,0,0] -> [3,3,0,0];
[1,C,1,0,0] -> [3,3,1,0];ICLK
[1,C,1,0,0] -> [7,7,1,1];
[1,C,1,0,0] -> [7,7,1,1];
[1,C,1,0,0] -> [7,7,1,1];
[1,C,1,0,0] -> [7,7,1,1];
[1,C,0,0,0] -> [7,7,0,1];
[1,C,0,0,0] -> [7,7,0,0];25
[1,C,0,0,0] -> [7,7,0,0];
[1,C,1,0,0] -> [7,7,1,0];ICLK
[1,C,1,0,0] -> [7,7,1,1];
[1,C,1,0,0] -> [7,7,1,1];
[1,C,1,0,1] -> [7,0,1,1];
[1,C,1,0,0] -> [7,0,1,1];
[1,C,1,1,0] -> [0,0,1,1];
[1,C,0,0,0] -> [0,0,0,1];
[1,C,0,0,0] -> [0,0,0,0];

end CLKMON

```
module TXMEMAD

title TRANSMIT MEMORY ADDRESS COUNTER  -- ATCINT U61
Leo J. Wapelhorst---FAA Tech Center-----6-19-90
   U61 device P22V10;

This module contains the Transmit Memory address counter.
It is a 9 bit counter which is preset with the # of words
to be transmitted. It produces an output when it reaches
zero.

CLK,En_Memad,Clr_Memad                          pin 1,2,13;
Q0,Q1,Q2,Q3,Q4                                  pin 23,22,21,20,19;
Q5,Q6,Q7,Q8,Q9                                  pin 18,17,16,15,14;

H,L,X,Z,C = 1, 0, .X., .Z., .C.;
COUNT  = [Q8..Q0];
COUNT_re = [Q9.re,Q8.re,Q7.re,Q6.re,Q5.re,Q4.re,Q3.re,Q2.re,Q1.re,Q0.re];
equations

COUNT  := (((COUNT+1) & En_Memad) # COUNT & !En_Memad);
COUNT_re = !Clr_Memad;

test_vectors

([CLK,En_Memad,Clr_Memad] ->  [COUNT])

[C,0,1] -> [^h00];
@radix 16;
@const n = 1;
@repeat 400
        {[C,1,1] -> [^h0 + n];
         [C,0,1] -> [^h0 + n];
         [C,0,1] -> [^h0 + n];
@const n = n+1;}
@radix ^hA;

end TXMEMAD
```

*U.S. GOVERNMENT PRINTING OFFICE 1991-534-[illegible]